SAND REPORT

SAND2002-3790
Unlimited Release
Printed November 2002

Xyce™ Parallel Electronic Simulator

User's Guide, Version 1.0.1

Scott A. Hutchinson, Eric R. Keiter, Robert J. Hoekstra, Lon J. Waters, Thomas V. Russo, Eric L. Rankin and Steven D. Wix

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/ordering.htm

SAND2002-3790
Unlimited Release
Printed November 2002

Xyce™ Parallel Electronic Simulator

User's Guide, Version 1.0.1

Scott A. Hutchinson, Eric R. Keiter, Robert J. Hoekstra, Eric L. Rankin
Computational Sciences

Lon J. Waters, Thomas V. Russo and Steven D. Wix
Component Information and Models

February 6, 2003

Abstract

This manual describes the use of the **Xyce** Parallel Electronic Simulator code for simulating electrical circuits at a variety of abstraction levels. The **Xyce** Parallel Electronic Simulator has been written to support, in a rigorous manner, the simulation needs of the Sandia National Laboratories electrical designers. As such, the development has focused on improving the capability over the current state-of-the-art in the following areas:

- Capability to solve extremely large circuit problems by supporting large-scale parallel computing platforms (up to thousands of processors). Note that this includes support for most popular parallel and serial computers.
- Improved performance for all numerical kernels (e.g., time integrator, nonlinear and linear solvers) through state-of-the-art algorithms and novel techniques.
- A client-server or multi-tiered operating model wherein the numerical kernel can operate independently of the graphical user interface (GUI).
- Object-oriented code design and implementation using modern coding-practices that ensure that the **Xyce** Parallel Electronic Simulator will be maintainable and extensible far into the future.

The code is a parallel code in the most general sense of the phrase - a message passing parallel implementation - which allows it to run efficiently on the widest possible

number of computing platforms. These include serial, shared-memory and distributed-memory parallel as well as heterogeneous platforms. Furthermore, careful attention has been paid to the specific nature of circuit-simulation problems to ensure that optimal parallel efficiency is achieved even as the number of processors grows.

Another feature required by designers is the ability to add device models, many specific to the needs of Sandia, to the code. To this end, the device package in the **Xyce** Parallel Electronic Simulator is designed to support a variety of device model inputs. These input formats include standard analytical models, behavioral models and look-up tables. Combined with this flexible interface is an architectural design that greatly simplifies the addition of circuit models.

One of the most important contribution **Xyce** makes to the designers at Sandia National Laboratories is in providing a platform for computational research and development aimed specifically at the needs of the Laboratory. With **Xyce**, Sandia now has an "in-house" capability with which both new electrical (e.g., device model development) and algorithmic (e.g., faster time-integration methods) research and development can be performed. Furthermore, these capabilities will then be migrated to the end users.

Acknowledgements

The authors would like to acknowledge the entire Sandia National Laboratories HPEMS (High Performance Electrical Modeling and Simulation) team, including Carolyn Bogdan, Regina Schells, Ken Marx, Steve Brandon, David Shirley and Bill Ballard, for their support on this project. We also appreciate very much the work of Becky Arnold and Mike Williamson for the help in reviewing this document.

Lastly, a very special thanks to Hue Lai for his help in typesetting this document in LaTeX.

Trademarks

Contacts

Bug Reports	`http://tvrusso.sandia.gov/bugzilla`
Email	`xyce-support@sandia.gov`
World Wide Web	`http://www.cs.sandia.gov/Xyce`

Contents

1. Preliminaries **15**
1.1 **Xyce** Overview 16
1.2 Installation 17
1.3 Quick Reference for Users of Other Circuit Codes 18
1.4 How to Use this Guide 18

2. Distinctive Features of Xyce **21**
2.1 **Xyce** Capabilities 22
2.2 Support for large-scale parallel computing 22
2.3 Analysis Support within **Xyce** 23

3. Simulation Examples with Xyce **25**
3.1 Example Circuit Construction 26
3.2 Running **Xyce** 29
Command Line Operation 29
3.3 DC Sweep Analysis 29
3.4 Transient Analysis 30

4. Simulation Design Creation **35**
4.1 Netlist Circuit Description 36
Netlist Overview 36
Netlist Elements 36
4.2 Devices Available for Simulation 38
Analog Devices 39
4.3 Parameters and Expressions 39
Parameters 39
How to Declare and Use Parameters 39
Expressions 41

5. Working with Models **45**
5.1 Definition of a Model 46
Defining models using model parameters 46
Defining models using subcircuit netlists 46
5.2 Model Organization 48
Model libraries 48
Model library configuration 49

5.3 Analog Behavioral Modeling 49
Overview of Analog Behavioral Modeling 49
Specifying ABM Devices 50

6. Creating and Running Analysis 51
6.1 Types of Analysis 52
6.2 Analysis Creation 52
6.3 Running a **Xyce** Simulation 53
Command Line Simulation 53
6.4 Parallel Partitioning Options 54
Chaco Static Partitioning of Circuit 55
Zoltan Partitioning of Linear System 55

7. DC Analysis 57
7.1 Overview of DC Sweep 58
7.2 Setting Up and Running a DC Sweep 58
7.3 OP Analysis 58

8. Transient Analysis 61
8.1 Transient Analysis Overview 62
8.2 Defining a Time-Dependent (transient) Source 62
Overview of Source Elements 62
Defining Transient Sources 62
8.3 Transient Calculation Time Steps 63
8.4 Checkpointing and Restarting 64
Checkpointing Command Format 64
Restarting Command Format 64

9. Results Output and Evaluation Options 67
9.1 Control of Results Output 68
`.PRINT` Command 68
9.2 Additional Output Options 68
`.OPTIONS OUTPUT` Command 68
9.3 Evaluating Solution Results 70

Appendix

A Netlist Reference 71
Netlist Commands 72
Analog Devices 91
B Command Line Arguments 149
C Runtime Environment 151
D Setting Convergence Parameters for **Xyce** 153
Adjusting Transient Analysis Error Tolerances 153
Adjusting Nonlinear Solver Parameters (in transient mode) 154
E Quick Reference for Orcad PSpice Users 155
GUI Support 155

Command Line Options 155
Device Support 155
Netlist Support 155
Converting PSpice ABM Models for Use in **Xyce** 156
F Quick Reference for ChileSPICE Users 159

Figures

3.1 Schematic of diode clipper circuit with DC and transient voltage sources. . . . 27
3.2 Diode clipper circuit netlist. 28
3.4 DC sweep voltages at Vin, node 2 and Vout. 30
3.3 Diode clipper circuit netlist for DC sweep analysis. 31
3.6 Sinusoidal input signal and clipped outputs. 32
3.5 Diode clipper circuit netlist for transient analysis. 33

5.1 Example subcircuit model. 47
5.2 Example subcircuit model. 48

6.2 Platform scripts for running **Xyce**. 54
6.1 Example netlist editing using XEmacs. 56

7.1 Diode clipper circuit netlist for DC sweep analysis. 59
7.2 DC sweep voltages at Vin, node 2 and Vout. 60

9.1 TecPlot plot of diode clipper circuit transient response from **Xyce** `.prn` file. . . 70

Tables

1.1 **Xyce** typographical conventions. 19

2.1 DC Analysis Capabilities. 24
2.2 Transient Analysis Capabilities. 24

3.1 DC Analysis References 30
3.2 Transient Analysis References. 34

4.1 Analog Devices References. 38
4.2 Analog Device Summary 40
4.3 Expression operators 43
4.4 Functions in arithmetic expressions 44

8.1 Summary of time-dependent sources supported by **Xyce**. 63

9.1 .PRINT command options. 69
A.6 Options for Device Package 81
A.7 Options for Time Integration Package. 84
A.8 Options for Nonlinear Solver Package. 86
A.9 Options for Nonlinear Solver Package under transient operation. 88
A.10 Options for Linear Solver Package. 89
A.11 Options for Parallel Support. 89
A.13 Capacitor Model Parameters. 94
A.15 Inductor Model Parameters. 97
A.17 Inductor Model Parameters. 99
A.19 Resistor Model Parameters. 100
A.21 Diode Model Parameters. 103
A.22 Diode Level 3 Model Parameters. 107
A.25 BJT Model Parameters. 123
A.27 MOSFET Model Parameters. 136
A.32 Voltage-controlled Switch Model Parameters. 142
A.34 Lossless (Ideal) Transmission Line Parameters. 144
A.36 PDE Device Instance Parameters. 148
A.37 PDE Device Model Parameters. 148
B.38 List of **Xyce** command line arguments. 149
C.39 Environment variables used to control **Xyce**. 151
C.40 Typical command lines for parallel execution. 152
F.41 Incompatibilities with ChileSPICE. 160

1. Preliminaries

Welcome to **Xyce**

The **Xyce** Parallel Electronic Simulator has been written to support, in a rigorous manner, the simulation needs of the Sandia National Laboratories electrical designers. It is targeted specifically to run on large-scale parallel computing platforms but also runs well on a variety of architectures including single processor workstations. It also aims to support a variety of devices and models specific to Sandia needs.

1.1 **Xyce** Overview

The **Xyce** Parallel Electronic Simulator development has focused on improving the capability over the current state-of-the-art in the following areas:

- Capability to solve extremely large circuit problems by supporting large-scale parallel computing platforms (up to thousands of processors). Note that this includes support for most popular parallel and serial computers.
- Improved performance for all numerical kernels (e.g., time integrator, nonlinear and linear solvers) through state-of-the-art algorithms and novel techniques.
- Support for modeling circuit phenomena at a variety of abstraction levels (device, analog, digital and mixed-signal) in a rigorous and tightly coupled manner, allowing for timely, full-system solutions.
- A client-server or multi-tiered operating model wherein the numerical kernel can operate distinctly from the simulation interface and the graphical user interface (GUI) (under development). This includes support for coupling with other simulation codes which may provide, for example, environmental information pertinent to the circuit (e.g, temperature as a function of time and space).
- Object-oriented code design and implementation using modern coding-practices that ensure that the **Xyce** Parallel Electronic Simulator will be maintainable and extensible far into the future.

The code is a parallel code in the most general sense of the phrase - a message passing parallel implementation - which allows it to run efficiently on the widest possible number of computing platforms. These include serial, shared-memory and distributed-memory parallel as well as heterogeneous platforms. Furthermore, careful attention has been paid to the specific nature of circuit-simulation problems to ensure that optimal parallel efficiency is achieved even as the number of processors grows.

As mentioned above, the **Xyce** Parallel Electronic Simulator is being developed in support of electrical designers of Sandia National Laboratory and, as such, is implementing several novel features that will make their job considerably easier. In addition to allowing the simulation of circuits of unprecedented size, **Xyce** includes novel approaches to numerical kernels such as time-stepping algorithms, nonlinear and linear solvers and improved device models. This approach aims to minimize the amount of simulation "tuning" required on the part of the designer and facilitate the code's successful usage.

Another feature of **Xyce** is the ability to add device models, many specific to the needs of Sandia, to the code. To this end, the device package in the **Xyce** Parallel Electronic Simulator is designed to support a variety of device model inputs. These input formats include standard analytical models, behavioral models and look-up tables, and support for

conductance values extracted from device-scale PDE models. Combined with this flexible interface is an architectural design that greatly simplifies the addition of circuit models.

For the user, this document contains a description of the **Xyce** Parallel Electronic Simulator, in which the following topics are specifically addressed. Chapters 2 and 3 give a simulation overview illustrating the distinctive features of **Xyce** and some examples of using the code. Chapters 4 through 5 describe how to create a basic circuit simulation design for **Xyce** using the standard netlist approach. Chapters 6 through 8 cover the creation and execution of circuit problems for **Xyce** on the supported platforms, including both serial and parallel architectures. This is followed by Chapter 9 that covers analyzing the results output by the code. Completing the document are the appendices including Appendix A, a netlist reference for the commands and elements supported within **Xyce** and Appendix B, which describes the available command line arguements for **Xyce**, as well as Appendices E and F, which contain quick-references for users of other circuit codes, such as Orcad's PSpice [1] and Sandia's ChileSPICE.

1.2 Installation

To obtain a copy of **Xyce**, contact the **Xyce** development team at one of the contacts given at `http://www.cs.sandia.gov/Xyce`. Once you have the distribution file, you install **Xyce** from the command line, following the instructions below. Examples are given for reference.

Instructions	Examples
Installation packages are named according the target operating system and architecture (parallel or serial).	`Install_Xyce_linux.tar.gz` (Linux Serial) `Install_Xyce_linux_MPI.tar.gz` (Linux Parallel) `Install_Xyce_mingw.zip` (Windows)
Unpack the appropriate package for your platform. A similarly named installation directory is then created. Windows users can unpack with programs such as *WinZip*, *PKZip*, *Winrar*, etc.	`$ gzip -d Install_Xyce_linux.tar.gz` `$ tar xf Install_Xyce_linux.tar`
Enter this directory and run the installation shell script. Windows users should run the `install.bat` batch file.	`$ cd Install_Xyce_linux` `$ sh install_linux.sh`
Provide the requested information.	`Where should Xyce be installed?` `/usr/local/Xyce-1.0`

Completing the steps above will unpack **Xyce** to the specified directory. **NOTE:** if installing *both* serial and parallel versions of **Xyce**, you must specify different directories for each in-

stallation location. Under the specified installation directories, the following subdirectories will be created:

- **bin** contains the executable used to start **Xyce**. The executable name will vary depending on the target operating system and architecture.
  - `runxyce` is the shell script for starting serial **Xyce** on Unix platforms.
  - `runxyce.bat` is the batch file for starting serial **Xyce** on Windows.
  - `xmpirun` is the wrapper script for mpirun used for running **Xyce** in parallel mode.
- **doc** contains the **Xyce** User's Guide and Release Notes. Read these for more information about this release and for detailed instructions on how to use **Xyce**.
- **lib** contains configuration files and libraries for **Xyce**.
- **test** contains sample netlists and verification tools.

1.3 Quick Reference for Users of Other Circuit Codes

Xyce is targeted at Sandia's designer community, many of whom have experience using other circuit codes, such as Orcad PSpice and Sandia's ChileSPICE. Much of the use of **Xyce** mirrors closely that in other circuit codes, but there are occasional differences that are documented in the appendices. Users of PSpice and ChileSPICE can get a "quick-start" by looking at Appendices E and F, respectively.

1.4 How to Use this Guide

This guide is designed so you can quickly find the information you need to use **Xyce**. It assumes that you are familiar with basic Unix-type commands, how Unix manages applications and files to perform routine tasks (e.g., starting applications, opening files and saving your work).

Typographical conventions

Before continuing in this User's Guide, it is important to understand the terms and typographical conventions used. Procedures for performing an operation are generally numbered with the following typographical conventions.

Notation	Example	Description
`Verbatim text`	`mpirun -np 2 Xyce`	Commands entered from the keyboard on the command line or text entered in a netlist.
Bold Roman Font	Set nominal temperature using the **TNOM** option.	SPICE-type parameters used in models, etc.
Gray Shaded Text	DEBUGLEVEL	Feature that is designed primarily for use by **Xyce** developers.
`[text in brackets]`	`Xyce [options] <netlist>`	Optional parameters.
`<text in angle brackets>`	`Xyce [options] <netlist>`	Parameters to be inserted by the user.
`<object with asterisk>*`	`K1 <ind. 1> [<ind. n>*]`	Parameter that may be multiply specified.
`<TEXT1|TEXT2>`	`.PRINT TRAN` `+ DELIMITER=<TAB|COMMA>`	Parameters that may only take specified values.

Table 1.1. Xyce typographical conventions.

2. Distinctive Features of **Xyce**

Chapter Overview

This chapter outlines the disctinctive features of **Xyce** that provide the user with leading edge capability in a variety of circuit simulation areas.

- Section 2.1, **Xyce** *Capabilities*
- Section 2.2, *Support for large-scale parallel computing*
- Section 2.3, *Analysis Support within* **Xyce**

2.1 **Xyce** Capabilities

The **Xyce** Parallel Electronic Simulator is a program that models circuit behavior at a variety of levels of fidelity - from Partial Differential Equations (PDE) based device models to analog to mixed signal and digital simulation. When used with the simulation front-end, **Xyce** will serve as an electronic laboratory for circuit design and optimization before any hardware is ever implemented. There are many tools available for basic circuit design. What sets **Xyce** apart are several key improvements over the state-of-the-art in circuit simulation as described in this chapter.

2.2 Support for large-scale parallel computing

Xyce is a truly parallel simulation code, designed and written from the ground up to support large-scale (up to thousands of processors) parallel computing architectures. This gives **Xyce** the capability to solve circuit problems of unprecedented size in time frames that make these simulations practical.

Xyce is a parallel code that uses a message passing parallel implementation, which allows it to run efficiently on the widest possible number of computing platforms. These include serial, shared-memory and distributed-memory parallel as well as heterogeneous platforms. Furthermore, careful attention has been paid to the specific nature of circuit-simulation problems to ensure that optimal parallel efficiency is achieved even as the number of processors grows (*parallel scaling*).

Improved performance for all numerical kernels

In writing **Xyce** from scratch, new algorithms and heuristics have been used which improve the overall performance of the numerical kernels for a given level of accuracy. As an example, several new nonlinear solution options are available which, when coupled with iterative linear solvers, can reduce execution time for many problems.

Ease of device model addition

Another feature of **Xyce** is the ability to add device models, many specific to the needs of Sandia, to the code. To this end, the device package[1] in **Xyce** has been designed with a flexible device-model interface that greatly simplifies the addition of circuit models. The device package interface has support for the standard "analog" or SPICE-type models as well as look-up tables, behavioral models and even PDE-based device models.

[1]The term "package" is a Unified Modeling Language (UML) term which refers to a group of classes, something akin to a module and is largely used in object-oriented programming.

Problem-specific modeling fidelity

Xyce has been designed to support the needs of Sandia National Laboratories' electrical designers who often need to model circuit phenomena at varying levels of fidelity - even within a given simulation. Thus, the code has been designed with an infrastructure that supports coupled simulation at several distinct abstraction levels: device, analog, digital and mixed-signal. While the code currently only supports analog simulation, development is proceeding to support both the device-scale[2] and digital/mixed-signal modeling. This is done in a rigorous and tightly coupled manner, allowing for timely, full-system solutions.

Analysis capability

Xyce currently supports DC and transient as well as a variety of optimization and design options available from the DAKOTA optimization framework [2] . This document does not cover the coupling of **Xyce** with DAKOTA. Sandia customers may contact the **Xyce** team for assistance with this "developing" capability. Full support is expected in the next major release.

Object-oriented code design and implementation

Xyce was designed and written from the ground up utilizing modern coding practices to ensure the optimal combination of code performance, code maintenance and code extensibility. This design allows for rapid implementation of new capability as well as long term maintenance of the code.

2.3 Analysis Support within **Xyce**

Several simulation analysis options are supported within **Xyce**. For basic analysis, **Xyce** currently supports DC and transient analysis; AC analysis intended to be supported in a future release. Also, a variety of optimization and design options are available via coupling with the DAKOTA optimization framework [2]. While DAKOTA is not distributed as part of **Xyce**, Sandia customers may contact the **Xyce** team for assistance with this capability.

DC Analysis

DC analysis is used to evaluate the circuit response to a direct current input source. DC analysis is automatically performed prior to a transient analysis simulation in order to provide initial conditions. This is commonly referred to as the "DC Operating Point" calculation. In addition, DC analysis can be performed to provide additional information about the circuit. Table 2.1 summarizes the various DC analysis types and corresponding **Xyce** implementation.

[2] The work on the device-scale coupling has been demonstrated but is not supported in the current release. It is expected to be made generally available by the next major release.

DC Analysis Type...	Xyce Calculates...
DC sweep	Steady-state voltages and currents when sweeping a source, a model parameter, or temperature over a range of values.
Operating Point	Steady-state initial conditions for voltages and currents at a DC bias.

Table 2.1. DC Analysis Capabilities.

Transient Calculations

The transient analysis capability within **Xyce** computes the transient performance of a circuit for a specified time interval. The initial conditions are provided by a DC analysis (DC Operating Point) automatically performed at the beginning of the transient simulation.

Transient Analysis Type...	Xyce Calculates...
Transient	Voltages and current tracked over time.

Table 2.2. Transient Analysis Capabilities.

Optimization and Design

The DAKOTA framework [2] provides a suite of optimization and design tools that can be used in conjunction with **Xyce**. For more information, contact a member of the **Xyce** team (see the Contact information on page 5.)

3. Simulation Examples with **Xyce**

Chapter Overview

This chapter provides an introduction to the process and tools used to generate and run circuit design simulations and to evaluate the results. An example circuit is provided for each available analysis type.

- Section 3.1, *Example Circuit Construction*
- Section 3.2, *Running* **Xyce**
- Section 3.3, *DC Sweep Analysis*
- Section 3.4, *Transient Analysis*

3.1 Example Circuit Construction

This section describes how to use **Xyce** to create the simple diode clipper circuit shown in Figure 3.1.

While a schematic edit and capture capability is under development, **Xyce** currently only supports circuit creation via netlist editing. **Xyce** supports most of the standard netlist entries common to Berkeley SPICE 3F5 and Orcad PSpice. For users who are familiar with PSpice netlists, the differences between PSpice and **Xyce** netlists are listed in Appendix E.

Example: diode clipper circuit

1. Open a new netlist file using a standard text editor (e.g., VI, Emacs, notepad, etc.).
2. Type the title on the first line of the netlist:

```
Diode Clipper Circuit
```

3. Create a 5V DC voltage source between nodes 1 and 0 by typing the following on a new line:

```
VCC 1 0 5V
```

4. Create another DC voltage source between nodes 3 and 0 by entering the following on a new line:

```
VIN 3 0 0V
```

5. Place the diodes in the circuit between nodes 2 and 1, and nodes 0 and 2, respectively, by entering the following lines:

```
D1 2 1 D1N3940
D2 0 2 D1N3940
```

6. Enter resistors R1, R2, R3 and R4, respectively:

```
R1 2 3 1K
R2 1 2 3.3K
R3 2 0 3.3K
R4 4 0 5.6K
```

7. Place the capacitor in the circuit:

```
C1 2 4 0.47u
```

8. Add the diode model to the netlist to complete it as Figure 3.2.

9. Complete the netlist by entering `.END` on the last line in the file. Save the file as `clipper.cir`. The complete netlist is shown in Figure 3.2 and the schematic in Figure 3.1.

The netlist in Figure 3.2 illustrates some of the syntax of a netlist input file. Netlists begin with a title (e.g., "`Diode Clipper Circuit`"), support comments (lines beginning with the "`*`" character), devices, model definitions and the "`.END`" statement.

This netlist file is not yet complete and will not run properly using **Xyce** (see Section 3.2 for instructions on running **Xyce**) as it lacks an analysis statement. As you proceed in this chapter, you will see how to add the appropriate analysis statement and run the clipper circuit.

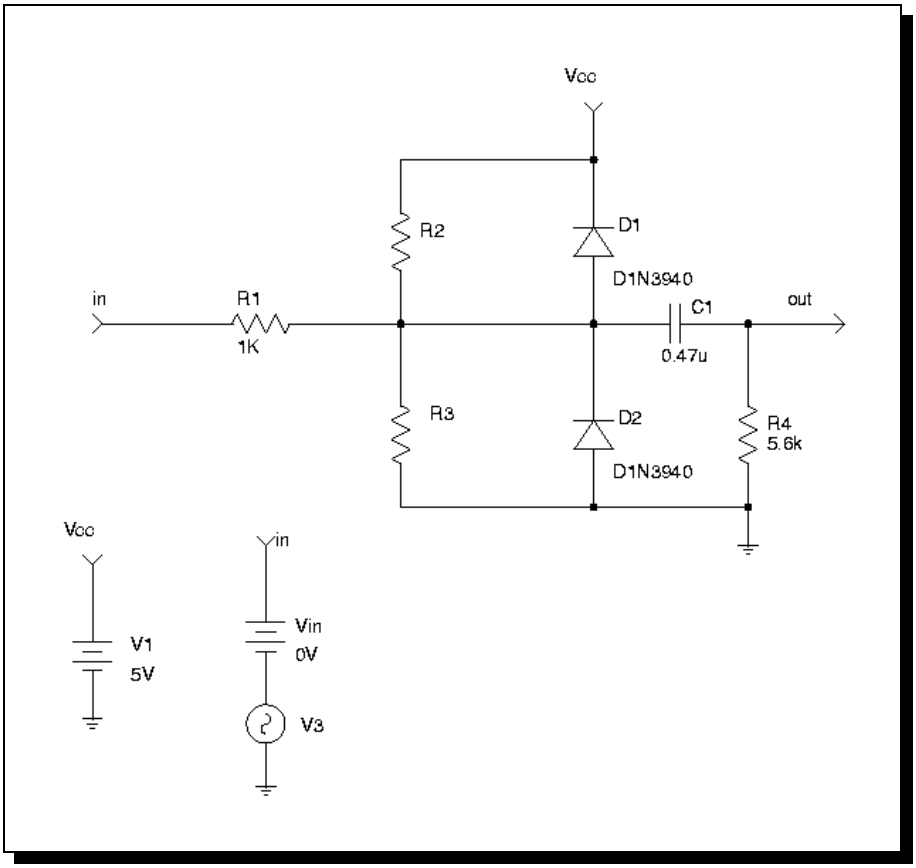

Figure 3.1. Schematic of diode clipper circuit with DC and transient voltage sources.

```
Diode Clipper Circuit
*
* Voltage Sources
VCC 1 0 5V
VIN 3 0 0V
* Diodes
D1 2 1 D1N3940
D2 0 2 D1N3940
* Resistors
R1 2 3 1K
R2 1 2 3.3K
R3 2 0 3.3K
R4 4 0 5.6K
* Capacitor
C1 2 4 0.47u
*
* GENERIC FUNCTIONAL EQUIVALENT = 1N3940
* TYPE:  DIODE
* SUBTYPE:  RECTIFIER
.MODEL D1N3940 D(
+          IS = 4E-10
+          RS = .105
+           N = 1.48
+          TT = 8E-7
+         CJO = 1.95E-11
+          VJ = .4
+           M = .38
+          EG = 1.36
+         XTI = -8
+          KF = 0
+          AF = 1
+          FC = .9
+          BV = 600
+         IBV = 1E-4)
*
.END
```

Figure 3.2. Diode clipper circuit netlist.

3.2 Running **Xyce**

While a GUI for **Xyce** is under development, **Xyce** can currently only be run from the command line. This section provides an overview of how to run **Xyce**.

Command Line Operation

Running **Xyce** from the command line is straightforward. The scripts `xmpirun` and `runxyce` created during Installation (1.2) set up the runtime environment for you and execute **Xyce**. Depending on whether you are using a version compiled with MPI or a serial version, there are two simple ways to have the computer begin running **Xyce**:

- Running serial **Xyce**:

```
> runxyce [options] <netlist filename>
```

- Running **Xyce** in parallel:

```
> xmpirun -np <# procs> [options] <netlist filename>
```

While **Xyce** is running, the progress of the simulation is output to the command line window.

For a more complete description of running **Xyce** in serial and in parallel, see Section 6.3.

3.3 DC Sweep Analysis

This section illustrates how to run a DC sweep analysis using **Xyce**. In this example we examine the DC response of the clipper circuit by running a DC sweep of the input voltage source (`Vin`) and reviewing at the results generated by **Xyce**. This example demonstrates using DC sweep analysis parameters that vary `Vin` from -10 to 15 volts in 1 volt steps.

Example: DC sweep analysis

To set up and run a DC sweep analysis using the diode clipper circuit:

1. Open the diode clipper circuit netlist file (`clipper.cir`) using a standard text editor (e.g., VI, Emacs, notepad, etc.).
2. Enter the analysis control statement in the netlist:

```
.DC VIN -10 15 1
```

3. Enter the output control statement:

```
.PRINT DC V(3) V(2) V(4)
```

4. Save the netlist file and run **Xyce** on the circuit. For example, to run serial **Xyce**:

```
> runxyce clipper.cir
```

5. Open the results file (clipper.cir.prn) and examine (or plot) the output voltages that were selected for nodes 3 (Vin), 2 and 4 (Out). Figure 3.4 shows the output plotted as a function of the swept variable Vin.

The modified netlist is shown below in Figure 3.3.

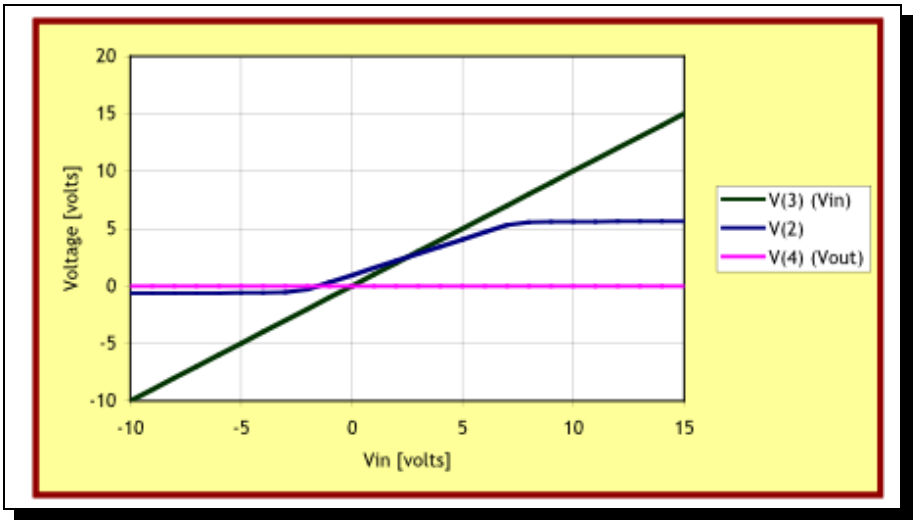

Figure 3.4. DC sweep voltages at Vin, node 2 and Vout.

Table 3.1 gives references for further explanation of the supported DC sweep analysis.

To find out more about. . .	See. . .
DC analysis for analog designs	Chapter 7, DC Analysis

Table 3.1. DC Analysis References

3.4 Transient Analysis

This section shows how to run a transient analysis using **Xyce**. In this example, we look at the transient response of the clipper circuit to a sinusoidal input voltage source (`Vin`)

```
Diode Clipper Circuit with DC sweep analysis statement
*
* Voltage Sources
VCC 1 0 5V
VIN 3 0 0V
* Analysis Command
.DC VIN -10 15 1
* Output
.PRINT DC V(3) V(2) V(4)
* Diodes
D1 2 1 D1N3940
D2 0 2 D1N3940
* Resistors
R1 2 3 1K
R2 1 2 3.3K
R3 2 0 3.3K
R4 4 0 5.6K
* Capacitor
C1 2 4 0.47u
*
* GENERIC FUNCTIONAL EQUIVALENT = 1N3940
* TYPE:  DIODE
* SUBTYPE:  RECTIFIER
.MODEL D1N3940 D(
+          IS = 4E-10
+          RS = .105
+           N = 1.48
+          TT = 8E-7
+         CJO = 1.95E-11
+          VJ = .4
+           M = .38
+          EG = 1.36
+         XTI = -8
+          KF = 0
+          AF = 1
+          FC = .9
+          BV = 600
+         IBV = 1E-4)
*
.END
```

Figure 3.3. Diode clipper circuit netlist for DC sweep analysis.

and review the results generated by **Xyce**. This example utilizes a sinusoidal input voltage source running at a frequency of 1 kHz and amplitude of 10 volts. To set up this example, we must modify the netlist to include this source.

Example: transient analysis

To set up and run a transient analysis using the diode clipper circuit:

1. Open the diode clipper circuit netlist file file (clipper.cir) using a standard text editor (e.g., VI, Emacs, notepad, etc.).
2. If you added DC analysis statements in the previous example, remove them (see Figure 3.4).
3. Enter the analysis control in the netlist:

```
.TRAN 2ns 2ms
```

4. Modify the input voltage source (`Vin`) to generate the sinusoidal input signal:

```
VIN 3 0 SIN(10V 1kHz)
```

5. Save the netlist file and run **Xyce** on the circuit. For example, to to run serial **Xyce**:

```
> runxyce clipper.cir
```

6. Open the results file and examine (or plot) the output voltages for nodes 3 (`Vin`), 2 and 4 (`Out`). The plot in Figure 3.6 shows the output plotted as a function of time.

The modified netlist is shown in Figure 3.5.

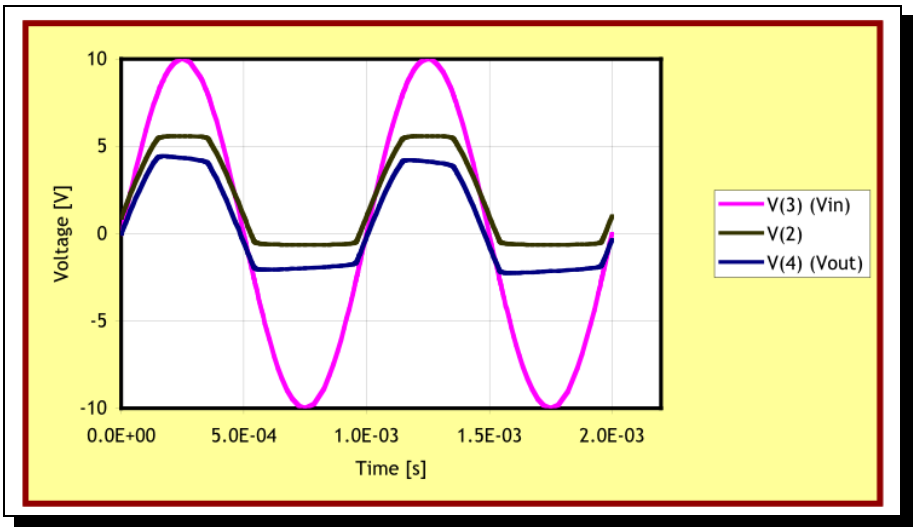

Figure 3.6. Sinusoidal input signal and clipped outputs.

```
Diode Clipper Circuit with transient analysis statement
*
* Voltage Sources
VCC 1 0 5V
VIN 3 0 SIN(0V 10V 1kHz)
* Analysis Command
.TRAN 2ns 2ms
* Output
.PRINT TRAN V(3) V(2) V(4)
* Diodes
D1 2 1 D1N3940
D2 0 2 D1N3940
* Resistors
R1 2 3 1K
R2 1 2 3.3K
R3 2 0 3.3K
R4 4 0 5.6K
* Capacitor
C1 2 4 0.47u
*
* GENERIC FUNCTIONAL EQUIVALENT = 1N3940
* TYPE:  DIODE
* SUBTYPE:  RECTIFIER
.MODEL D1N3940 D(
+          IS = 4E-10
+          RS = .105
+           N = 1.48
+          TT = 8E-7
+         CJO = 1.95E-11
+          VJ = .4
+           M = .38
+          EG = 1.36
+         XTI = -8
+          KF = 0
+          AF = 1
+          FC = .9
+          BV = 600
+         IBV = 1E-4)
*
.END
```

Figure 3.5. Diode clipper circuit netlist for transient analysis.

Table 3.2 below gives references for further explanation of the supported transient analysis.

To find out more about. . .	See. . .
Transient analysis for analog designs	Chapter 8, Transient Analysis

Table 3.2. Transient Analysis References.

4. Simulation Design Creation

Chapter Overview

This chapter contains basic information on creating circuit designs. Sections include:

- Section 4.1 *Netlist Circuit Description*
- Section 4.2 *Devices Available for Simulation*
- Section 4.3 *Parameters and Expressions*

4.1 Netlist Circuit Description

Netlist Overview

Using a netlist to describe a circuit for **Xyce** is the primary method used for running a circuit simulation. Netlist support within **Xyce** largely conforms to that used by Berkeley SPICE 3F5 with several new options for controlling functionality unique to **Xyce**. In a netlist, the circuit is described by a set of "element lines" which define the circuit elements and their values, the circuit topology (the connection of the circuit elements), and a variety of control options for the simulation. The first line in the netlist file must be a title and the last line must be "`.END`". Between these two constraints, the order of the statements is irrelevant.

Netlist Elements

An "element line", for which the format is determined by the specific element type, defines each circuit element instance. The general format is given by:

```
<type><name> <node information> <element information...>
```

The `<type>` must be a letter (A through Z) and the `<name>` follows immediately. For example, `RARESITOR` specifies a type=resistor with a name `ARESITOR`. Fields on a line are separated by spaces, commas, an equal sign or a left or right parenthesis.

A number field may be an integer or a floating-point value. Either one may be followed by one of the following scaling factors:

Symbol	Equivalent Value
T	10^{12}
G	10^{9}
Meg	10^{6}
K	10^{3}
mil	25.4^{-6}
m	10^{-3}
u (μ)	10^{-6}
n	10^{-9}
p	10^{-12}
f	10^{-15}

Node information is given in terms of node names, which are arbitrary character strings. The only requirement is that the ground node is named '0'. There are some restrictions on the circuit topology:

- There can be no loop of voltage sources and/or inductors.
- There can be no cut-set of current sources and/or capacitors.
- Every node must have a DC path to ground.
- Every node must have at least two connections (with the exception of unterminated transmission lines and MOSFET substrate nodes).

The following line provides an example of an element line that defines a resistor between nodes 1 and 3 with a resistance value of $10\ k\Omega$.

Example: `RARESISTOR 1 3 10K`

Title, Comments and End

The title line is required to be the first line in the input netlist and is included in the output file.

Example: `Test RLC Circuit`

The "`.End`" line must be the last line in the netlist.

Example: `.END`

Comments are supported in netlists and are indicated by placing an asterisk at the beginning of the comment line. They may occur anywhere in the netlist *but* they must be at the beginning of a line. **Xyce** also supports "in-line" comments. An in-line comment is designated by a leading semicolon and may occur on any line. Everthing after the semicolon is taken as a comment and ignored. Any line that begins with leading white space is also considered to be a comment.

Example: `* This is a netlist comment.`

Example: `.DC .... * This type of inline comment is not supported.`

Example: `.DC .... ; This type of inline comment is supported.`

Netlist Commands

Command elements are used to describe the analysis being defined by the netlist. Examples include analysis types, initial conditions, device models and output control. Section A in Appendix A serves as a reference for these commands.

Example: `.PRINT TRAN V(Vout)`

Analog Devices

The analog devices supported include most of the standard circuit components normally found in circuit simulators such as SPICE 3F5, PSpice, etc., plus several Sandia specific devices.

Example: `D_CR303 N_0065 0 D159700`

Table 4.1 below gives references for further explanation of the supported analog devices.

To find out more about...	See...
Analog devices	Subsection A in Appendix A, Analog Devices

Table 4.1. Analog Devices References.

4.2 Devices Available for Simulation

This section describes the different types of analog devices supported in **Xyce**. These include standard analog devices, sources (dependent and independent) and subcircuits. Each device description has the following information:

- A description and an example of the netlist syntax
- The corresponding model types and descriptions, where applicable
- The corresponding lists of model parameters and descriptions, where applicable
- The associated circuit diagram and model equations (as necessary)

These analog devices include all of the standard circuit components needed for most analog circuits. User defined models may also be implemented using the `.MODEL` (model definition) statement and macromodels as subcircuits using the `.SUBCKT` (subcircuit) statement.

Analog Devices

Xyce supports many analog devices, including sources, subcircuits and behavioral models. The devices are classified into device types, each of which can have one or more model types. For example, the BJT device type has two model types: NPN and PNP.

The device element statements in the netlist always start with the name of the individual device instance. The first letter of the name determines the device type. The format of the following information depends on the device type and its parameters. The Device Type summary table (Table 4.2) lists all of the analog devices supported by **Xyce**. Each standard device is then described in more detail in the following sections. Except where noted, the devices are based upon those found in [3].

Table 4.2 is a summary of the analog device types and the form of their netlist formats:

4.3 Parameters and Expressions

In addition to explicit values, the user may use parameters and expressions to symbolize numeric values in the circuit design.

Parameters

A parameter is like a programming variable that represents a numeric value by name. Once you have defined a parameter (declared its name and given it a value) at a particular level in the circuit hierarchy, you can use it to represent circuit values at that level or any level directly beneath it in the circuit hierarchy. One way that you can use parameters is to apply the same value to multiple part instances.

How to Declare and Use Parameters

In order to use a global parameter in a circuit, one must:

- define the parameter using a `.PARAM` statement within a netlist
- replace an explicit value with the parameter in the circuit

Note that **Xyce** reserves several keywords that may not be used as parameter names. These are:

- `Time`
- `Vt`

Device Type	Designator Letter	Typical Netlist Format[a]
Capacitor	C	C<name> <+ node> <- node> [model name] <value> + [IC=<initial value>]
Inductor	L	L<name> <+ node> <- node> [model name] <value> + [IC=<initial value>]
Resistor	R	R<name> <+ node> <- node> [model name] <value> + [L=<length>] [W=<width>]
Diode	D	D<name> <anode node> <cathode node> + <model name> [area value]
Mutual Inductor	K	K<name> <inductor 1> [<ind. n>*] + <linear coupling or model>
Independent Voltage Source	V	V<name> <+ node> <- node> [[DC] <value>] + [AC <magnitude value> [phase value]] + transient specification]
Independent Current Source	I	I<name> <+ node> <- node> [[DC] <value>] + [AC <magnitude value> [phase value]] + [transient specification]
Voltage Controlled Voltage Source	E	E<name> <+ node> <- node> <+ controlling node> + <- controlling node> <gain>
Voltage Controlled Current Source	G	G<name> <+ node> <- node> <+ controlling node> + <- controlling node> <transconductance>
Nonlinear Dependent Source (B Source)	B	B<name> <+ node> <- node> <I or V>={<expression>}
Bipolar Junction Transistor (BJT)	Q	Q<name> <collector node> <base node> <emitter node> [substrate node] <model name> [area value]
MOSFET	M	M<name> <drain node> <gate node> <source node> + <bulk/substrate node> <model name> + [common model parameter]*
Transmission Line	T	T<name> <A port + node> <A port - node> + <B port + node> <B port - node> + <ideal specification>
Voltage Controlled Switch	S	S<name> <+ switch node> <- switch node> + <+ controlling node> <- controlling node> + <model name>
Subcircuit	X	X<name> [node]* <subcircuit name> + [PARAMS:[<name>=<value>]*]
PDE Devices	Z	Z<name> <node1> <node2> [node3] + [node4] <model name>

Table 4.2. Analog Device Summary

[a] For a more complete description of the syntax for supported devices, see Appendix A

- Temp
- GMIN

However, in this first release of **Xyce**, only Time is predefined.

Example: Declaring a parameter

1. Locate the level in the circuit hierarchy at which the .PARAM statement declaring a parameter will be placed (note: a global parameter that can be used anywhere in the netlist can be declared by placing the .PARAM statement at the top-most level of the circuit).
2. Name the parameter and give it a value. The value can be numeric or given by an expression:

```
.SUBCKT subckt1 n1 n2 n3
.PARAM res = 100
*
* other netlist statements here
*
.ENDS
```

3. Note: the parameter "res" can be used anywhere within the subcircuit subckt1 including subcircuits defined within it, but cannot be used outside of subckt1.

Example: Using a parameter in the circuit

1. Find the numeric value that is to be replaced by a parameter: a device instance parameter value, model parameter value, etc. The value being replaced must be accessible with the current hierarchy level.
2. Replace the numeric value with the parameter name contained within braces ({}) as in:

```
R1 1 2 {res}
```

Expressions

In **Xyce**, an expression is a mathematical relationship that may be used any place one would use a number (numeric or boolean). **Xyce** evaluates the expression to a value when it reads in a new circuit netlist.

To use an expression in a circuit netlist:

1. Locate the value to be replaced (component, model parameter, etc.).
2. Substitute the value with an expression utilizing the $\{\}$ syntax:

   $\{expression\}$

   where *expression* can contain any of the following:

   - available operators from those in Table 4.3
   - included functions from those in Table 4.4
   - user-defined functions
   - the system variable `TIME` for use only in ABM expressions, see Chapter5.3
   - user-defined parameters that are within scope
   - literal operands

   The braces ($\{\}$) instruct **Xyce** to evaluate the expression and use the resulting value.

Example: Using an expression

Scaling the DC voltage of a $12V$ independent voltage source, designated `VF`, by some factor can be accomlished by the following netlist statements (in this example the factor is 1.5):

```
.PARAM FACTORV=1.5
VF 3 4 {FACTORV*12}
```

Xyce will evaluate the expression to:

`12 * 1.5` or `18` volts

[1] Logical and relational operators are used only with the `IF()` function.

Class of operator...	Operator...	Meaning
arithmetic	+	addition or string concatenation
	-	subtraction
	*	multiplication
	/	division
	**	exponentiation
logical[1]	~	unary NOT
	|	boolean OR
	^	boolean XOR
	&	boolean AND
relational	==	equality
	!=	non-equality
	>	greater-than
	>=	greater-than or equal
	<	less-than
	<=	less-than or equal

Table 4.3. Expression operators

Function...	Meaning...	Explanation...
ABS(x)	$\|x\|$	
SQRT(x)	$\sqrt{x}$	
EXP(x)	e^x	
LN(x)	$\ln(x)$	log base e
LOG(x)	$\log(x)$	log base 10
SIN(x)	$\sin(x)$	x in radians
ASIN(x)	$\arcsin(x)$	result in radians
SINH(x)	$\sinh(x)$	x in radians
ASINH(x)	$\sinh^{-1}(x)$	result in radians
COS(x)	$\cos(x)$	x in radians
ACOS(x)	$\arccos(x)$	result in radians
COSH(x)	$\cosh(x)$	x in radians
ACOSH(x)	$\cosh^{-1}(x)$	result in radians
TAN(x)	$\tan(x)$	x in radians
ATAN(x)	$\arctan(x)$	result in radians
TANH(x)	$\tanh(x)$	x in radians
ATANH(x)	$\tanh^{-1}(x)$	result in radians
ATAN2(x,y)	$\arctan(y/x)$	result in radians
SGN(x)	+1 if $x > 0$ 0 if $x = 0$ -1 if $x < 0$	
U(x)	1 if $x > 0$ 0 otherwise	suppress a value until a given time
URAMP(x)	x if $x > 0$ 0 otherwise	
IF(t,x,y)	x if t is true, y otherwise	t is an expression using the relational operators in Table 4.3
DDT(x)	time derivative of x	
SDT(x)	time integral of x	

Table 4.4. Functions in arithmetic expressions

5. Working with Models

Chapter Overview

This chapter contains model ideas and a summary of the ways to create and modify models. Sections include:

- Section 5.1, *Definition of a Model*
- Section 5.2, *Model Organization*
- Section 5.3, *Analog Behavioral Modeling*

5.1 Definition of a Model

A model describes the electrical performance of a *part*. A part is a component in the circuit with *specific simulation properties* that *define* the part. In a netlist, a part is identified by its implementation properties designated by the associated model name.

Depending on the given device type, a model is defined as either:

- a model parameter set
- a subcircuit netlist

Both methods of defining a model use a netlist format, with precise syntax rules as described below.

Defining models using model parameters

Xyce currently has no built-in models. However, models can be defined for a device by changing some or all of the *model parameters* from their defaults via the `.MODEL` syntax. For example:

Example: `.MODEL MLOAD1 NMOS (LEVEL=2 VTO=0.5 CJ=0.025pF)`

Defining models using subcircuit netlists

In **Xyce**, models may also be defined using the *subcircuit syntax:* `.SUBCKT`/`.ENDS`. This syntax includes:

- *netlists* to define the configuration and function of the part.
- *variable input parameters* to modify the model.

See Figure 5.1 for an example.

```
*** SUBCIRCUIT: l3dsc1
*** Parasitic Model: microstrip
*** Only one segment
.SUBCKT l3dsc1 1 3 2 4
C01 1 0 4.540e-12
RG01 1 0 7.816e+03
L1 1 5 3.718e-08
R1 5 2 4.300e-01
C1 2 0 4.540e-12
RG1 2 0 7.816e+03
C02 3 0 4.540e-12
RG02 3 0 7.816e+03
L2 3 6 3.668e-08
R2 6 4 4.184e-01
C2 4 0 4.540e-12
RG2 4 0 7.816e+03
CM012 1 3 5.288e-13
KM12 L1 L2 2.229e-01
CM12 2 4 5.288e-13
.ENDS
```

Figure 5.1. Example subcircuit model.

Subcircuit Hierarchy

Xyce supports the definition of subcircuits within other subcircuits. Each subcircuit definition introduces a new level in the circuit hierarchy with the top level begin the main circuit. If a second level is defined, it is composed of the subcircuits in the main circuit and each subsequent level is composed of the subcircuits contained in the previous level. A subcircuit may also contain other definitions such as models via the `.MODEL` statement, parameters via the `.PARAM` statement, and functions via the `.FUNC` statement.

In this context, the subcircuit defines the "scope" for the definitions it contains. That is, *the definitions contained within a subcircuit can be used within that subcircuit and/or within any subcircuit it contains*. Any definitions occuring in the main circuit have global scope and can be used anywhere in the circuit. A name, such as a model, parameter, function or subcircuit name, occurring in a definition at one level of a circuit hierarchy can be redefined at any lower level contained directly by the subcircuit. In this case, the new definition applies at the given level and those below.

In the following example, the model named `MOD1` can be used in subcircuits `SUB1` and `SUB2`

but not in the subcircuit SUB3. The parameter P1 has a value of 10 in subcircuit SUB1 and a value of 20 in subcircuit SUB2.

```
.SUBCKT SUB1 1 2 3 4
.MODEL MOD1 NMOS(LEVEL=2)
.PARAM P1=10
*
* subcircuit devices omitted for brevity
*
.SUBCKT SUB2 1 3 2 4
.PARAM P1=20
*
* subcircuit devices omitted for brevity
*
.ENDS
.ENDS

.SUBCKT SUB3 1 2 3 4
*
* subcircuit devices omitted for brevity
*
.ENDS
```

Figure 5.2. Example subcircuit model.

5.2 Model Organization

The organization of models entails the following fundamental concepts:

- model definitions are saved in model library files.
- model libraries must be constructed so that **Xyce** will look to them for model definitions.

Model libraries

Device model and subcircuit definitions are organized into model libraries. These libraries are text files (similar to netlist files) that have one or more model definitions. Model library names usually end with a .lib extension.

As a rule-of-thumb, model libraries files typically include similar model types. In these files, the *header comments* describe the models therein.

Model library configuration

Xyce searches the model library files for the model names given by the `.MODEL` statement for parts in the netlist. It then uses these model definitions in the simulation. For **Xyce** to be able to find these model definitions, the libraries must be "included" in the netlist. This is accomplished by adding a `.INCLUDE` statement to the netlist. As an example:

```
*** SUBCIRCUIT: l3dsc1
*** Parasitic Model: microstrip
*** Only one segment
.SUBCKT l3dsc1 1 3 2 4
.INCLUDE "model.lib"
C01 1 0 4.540e-12
RG01 1 0 7.816e+03
L1 1 5 3.718e-08
R1 5 2 4.300e-01
C1 2 0 4.540e-12
RG1 2 0 7.816e+03
C02 3 0 4.540e-12
RG02 3 0 7.816e+03
L2 3 6 3.668e-08
R2 6 4 4.184e-01
C2 4 0 4.540e-12
RG2 4 0 7.816e+03
CM012 1 3 5.288e-13
KM12 L1 L2 2.229e-01
CM12 2 4 5.288e-13
.ENDS
```

The scoping rules for `.INCLUDE` statements is the same as for other types of definitions as outlined in the preceding section.

5.3 Analog Behavioral Modeling

Overview of Analog Behavioral Modeling

The analog behavioral modeling capability of **Xyce** provides for flexible descriptions of electronic components in terms of a transfer function or lookup table. In other words,

a mathematical relationship is used to model a circuit segment removing the need for component by component design.

In **Xyce**, the `B` nonlinear dependent source device type is used for analog behavioral modeling . The PSpice equivalent is its `E` and `G` voltage controlled source device types. The conversion from a PSpice analog behavioral model will be described in a later section.

Specifying ABM Devices

ABM devices (`B` devices) are specified in a netlist the same way as other devices. Customizing the operational behavior of the device is achieved by defining an ABM expression describing how inputs are transformed into outputs.

Device and node names in ABM expressions

ABM expressions follow the same rules as other expressions in a netlist with the additional ability to specify signals (node voltages and voltage source currents) in the expression. In ABM expressions, refer to signals by name. **Xyce** recognizes the following constructs in ABM expressions:

- `V(<node name>)`
- `V(<node name>,<node name>)`
- `I(<voltage source name>)`

In a hierarchical circuit (a circuit with possibly nested levels of subcircuits), voltage source names that appear in an ABM expression must be the name of a voltage source in the same subcircuit as the ABM device. Similarly, node names in an ABM expression must be the node names of one or more devices in the same subcircuit as the ABM device.

6. Creating and Running Analysis

Chapter Overview

This chapter outlines methods for creating and running different types of circuit analysis using **Xyce**. This is given in the following sections:

- Section 6.1, *Types of Analysis*
- Section 6.2, *Analysis Creation*
- Section 6.3, *Running a* **Xyce** *Simulation*
- Section 6.4, *Parallel Partitioning Options*

6.1 Types of Analysis

Currently **Xyce** supports only DC and transient analysis with plans to support AC analysis and a variety of design analysis types (e.g. design optimization, sensitivity analysis) in the future.

DC Analysis

DC analysis is used to evaluate the circuit response to a direct current input source. DC analysis is automatically performed prior to a transient analysis simulation in order to provide initial conditions. In addition, DC analysis can be performed in a stand-alone manner to provide additional information about the circuit. Currently, only the DC sweep analysis is supported which calculates steady-state voltages and currents when sweeping a source or a model parameter.

Transient Calculations

The transient analysis capability within **Xyce** computes the transient performance of a circuit for a specified time interval. The initial conditions are provided by a DC analysis automatically performed at the beginning of the transient simulation. From this, **Xyce** computes the transient response from `TIME=0` to a specified time. During this calculation, the **Xyce** time integrating algorithms adjust the calculation's time step size to correspond to the circuit's response while minimizing the overall number of time steps. This is accomplished by ensuring the default or user specified error requirements while maximizing the time-step size at any point. During fast changing portions of the simulation, the time-step size is reduced, while during less active portions, the step-size is increased.

6.2 Analysis Creation

To set up a simulation analysis:

1. Open a new or existing netlist text file using a text editor (e.g., XEmacs, VI, Notepad). See Figure 6.1 for an example of editing a netlist using XEmacs.
2. Enter the netlist commands that describe the circuit, analysis type and output information. See Chapter 3 for a description on how to setup a circuit design.
3. Enter the required parameter values and other information to complete the analysis specifications.
4. Save the file for running in **Xyce**.

Specific information for setting up each type of analysis is discussed in the following chapters.

6.3 Running a **Xyce** Simulation

While a GUI for **Xyce** is under development and will be part of an overall simulation framework, **Xyce** can currently only be run from the command line. This section outlines the running of **Xyce** for both serial and MPI parallel simulations.

Command Line Simulation

Running **Xyce** from the command line is straightforward. The scripts `xmpirun` and `runxyce` created during installation (see Section 1.2) set up the runtime environment and execute **Xyce**. (Microsoft Windows users should use the `runxyce.bat` batch file.) Depending on whether you are using a version compiled with MPI support or a serial version, there are two ways, respectively, to begin running **Xyce**:

- Running serial **Xyce**:

```
> runxyce [options] <netlist filename>
```

- Running **Xyce** in parallel:

```
> xmpirun -np <# procs> [options] <netlist filename>
```

where [`options`] are the command line arguments for **Xyce**. For example, to log output to a file named `sample.log` and use faster direct matrix access type:

```
$ runxyce -l sample.log -dma on <netlist filename>
```

The next example runs parallel **Xyce** on four processors and places the results into a comma separated value file named `results.csv`:

```
$ xmpirun -np 4 -delim COMMA -o results.csv <netlist filename>
```

These examples assume that `<netlist filename>` is either in the current working directory or includes the path (full or relative) to the netlist file. See Appendix B for command line details. Help is also accessible with the `-h` option.

For MPI runs, [`options`] may also include command line arguments to `mpirun`. Consult the documentation installed with MPI on your platform for more details and MPI options. The "`-np <# procs>`" denotes the number of processors to use for the simulation for the particular computer one is using. *NOTE: It is critical that the number of processors used is less than the number of devices and voltage nodes in the netlist.* The appropriate script used to run **Xyce** for each supported platform is listed in the Table 6.2.

Computer Architecture	OS	Serial Executable	MPI Executable
Sun	Solaris 8.0	runxyce	Not Available
Compaq	DEC OSF		xmpirun
SGI 64 bit	IRIX 6.5		
Intel X86	Linux		
Intel X86	FreeBSD		
Intel X86	Microsoft Windows 2000	runxyce.bat	Not Available

Figure 6.2. Platform scripts for running **Xyce**.

While **Xyce** is running, the progress of the simulation is output to the command line window or optionally to a log file if so specified on the command line (see command line options in Appendix B).

6.4 Parallel Partitioning Options

For running on most platforms, the default procedure outlined above will be sufficient, although, to improve efficiency on parallel computers, the user can make use of graph partitioning methods to more effectively distribute the problem to the processors. **Xyce** currently has two graph partitioning options available. These partitioning utilities subdivide the circuit problem into sections that are then distributed to the nodes [1] (processors) on a parallel computer. A good partition can have a dramatic effect on the parallel performance of circuit simulation run. Basically, there are two key components to a good partition: 1) achieving an effective load balance and 2) minimizing communication overhead. An effective load balance ensures that the computational load of the calculation is equally distributed among the available processors. Minimizing communication overhead seeks to distribute the problem in a way that reduces the impact of underlying message passing during the simulation run. **Xyce** has integrated within it two partitioning libraries - the **Chaco** static partitioner and the **ZOLTAN** library of parallel partitioning heuristics.

[1] The term "node" is used here in the parallel computing context to refer to a unit of a parallel computer. This is a more general term the that of "processor"; in many distributed memory computers, two or more processors share a memory unit and are collectively referred to as a "node". Note also that the context should prevent confusion with "circuit nodes".

Chaco Static Partitioning of Circuit

Chaco is accessible using the `.OPTIONS PARALLEL` line in the netlist. By adding this line to the netlist, **Chaco** will be used to partition the initial circuit before it is distributed to processors. **Chaco** partitioning can be controlled through a `Chaco_User_Params` file that must be in the execution directory. See the **Chaco User Guide** [4] for details.

Currently, one parameter is available for **Chaco** partitioning: using `DISTRIBINDSRCNODES=1` as a parallel option can be very effective for an improved partitioning, especially for large digital circuits. However, using this option places certain restrictions on `RESTART` and `OUTPUT` options. Specifically, the restart functionality can only be used for an identical number of processors and an identical partitioning. Furthermore, electrical current cannot be output for the distributed voltage sources and voltage cannot be output for the distributed voltage nodes.

due to the renaming and distribution of some independent sources and their associated voltage nodes. Restarting can only be used for an identical number of processors and partitioning. Current cannot be output for the distributed voltage sources and voltage cannot be output for the distributed voltage nodes.

Zoltan Partitioning of Linear System

Zoltan is accessible through the `.OPTIONS LINEAR` control line in the netlist. By adding an option "`TR_LOADBALANCE=1`" to the linear options, the linear system is statically load balanced based on the graph of the Jacobian matrix. The local system is also reordered based on nested dissection which should improve conditioning and minimize fill. These techniques can be very effective for improving the efficiency of the iterative linear solvers.

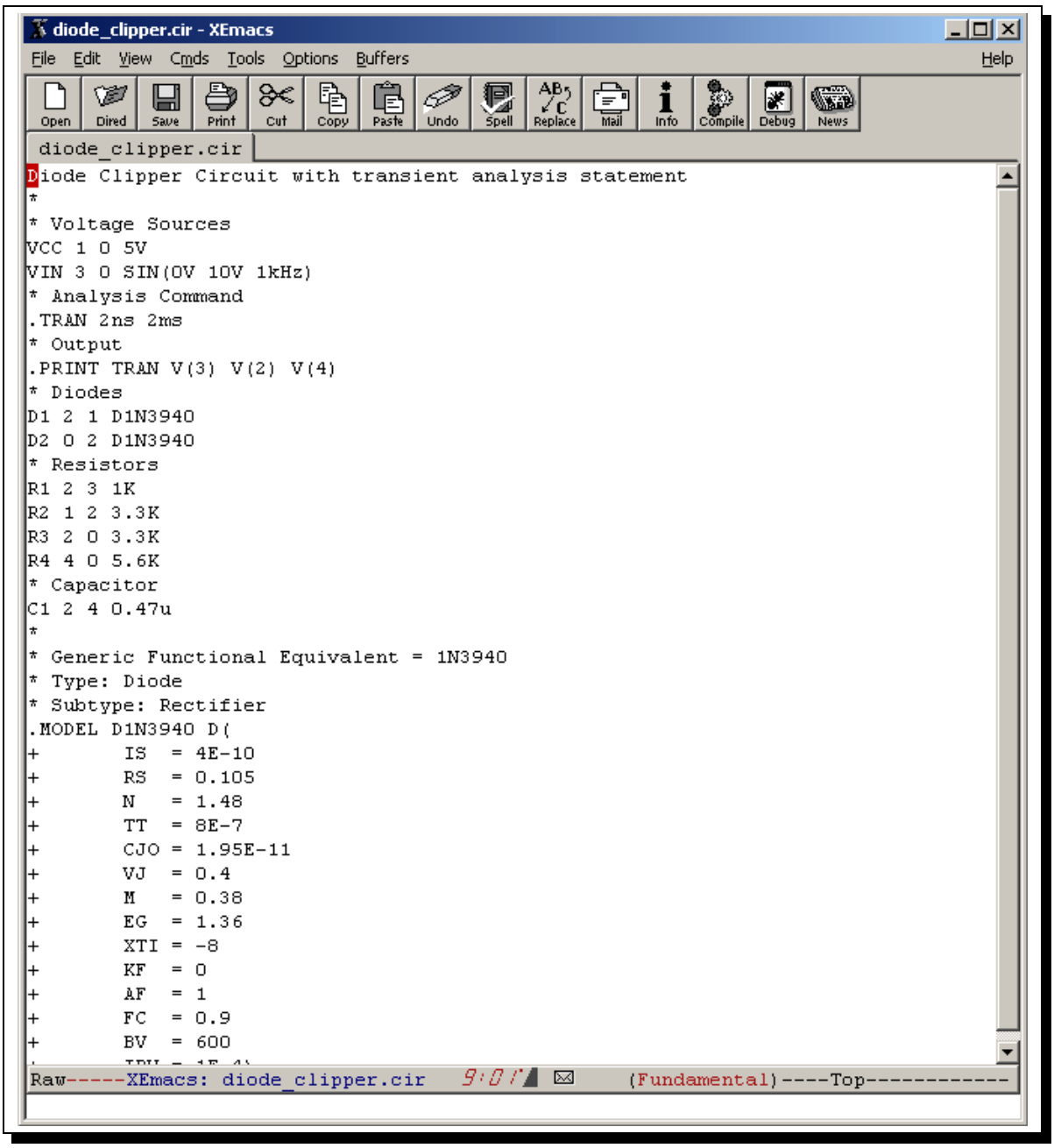

Figure 6.1. Example netlist editing using XEmacs.

7. DC Analysis

Chapter Overview

This chapter explains how to set up DC analysis and includes the following sections:

- Section 7.1, *Overview of DC Sweep*
- Section 7.2, *Setting Up and Running a DC Sweep*

7.1 Overview of DC Sweep

The DC sweep analysis capability in **Xyce** carries out a sweep, in DC mode, on a circuit. DC sweep is supported for a source (current or voltage), through a range of specified values. As the sweep proceeds, the bias point is computed for each value in the specified range of the sweep.

If the variable to be swept is a voltage or current source, a DC source must be used. The DC value is set in the netlist (see Appendix A). In simulating the DC response of an analog circuit, **Xyce** eliminates any time dependence from the circuit. This is accomplished by treating all capacitor elements as open circuits, all inductor elements as short circuits and using only the DC values of both voltage and current sources.

7.2 Setting Up and Running a DC Sweep

Following the example given in Section 3.3, the diode clipper circuit netlist is shown in Figure 7.1 with a DC sweep analysis specified. Here, the voltage source `Vin` is swept from -10 to 15 in 1 volt increments, resulting in 26 DC operating point calculations. Note also that the default setting for `Vin` is ignored during these calculations. All other source values use the specified values (`VCC = 5V` in this case).

Running **Xyce** on this netlist produces an output results file named `clipper.cir.prn`. Plotting this data produces the graph shown in Figure 7.2.

7.3 OP Analysis

Xyce also supports `.OP` analysis statements. In **Xyce**, `.OP` should be considered as a shorthand for a single step DC sweep, in which all the default operating point values are used. One can also consider `.OP` analysis to be the operating point calculation which would occur as the intitial step to a transient calculation, without the subsequent time steps.

This capability was mainly added so that the code would be able to handle legacy netlists which used this type of analysis statement. In most versions of SPICE, using `.OP` will result in extra output which is not available from a DC sweep. That additional output capability has not yet been implemented in **Xyce**.

```
Diode Clipper Circuit
** Voltage Sources
VCC 1 0 5V VIN 3 0 0V
* Analysis Command
.DC VIN -10 15 1
* Output
.PRINT DC V(3) V(2) V(4)
* Diodes
D1 2 1 D1N3940 D2 0 2 D1N3940
* Resistors
R1 2 3 1K
R2 1 2 3.3K
R3 2 0 3.3K
R4 4 0 5.6K
* Capacitor
C1 2 4 0.47u
** GENERIC FUNCTIONAL EQUIVALENT = 1N3940
* TYPE:  DIODE
* SUBTYPE:  RECTIFIER
.MODEL D1N3940 D(
+          IS = 4E-10
+          RS = .105
+           N = 1.48
+          TT = 8E-7
+         CJO = 1.95E-11
+          VJ = .4
+           M = .38
+          EG = 1.36
+         XTI = -8
+          KF = 0
+          AF = 1
+          FC = .9
+          BV = 600
+         IBV = 1E-4)
* .END
```

Figure 7.1. Diode clipper circuit netlist for DC sweep analysis.

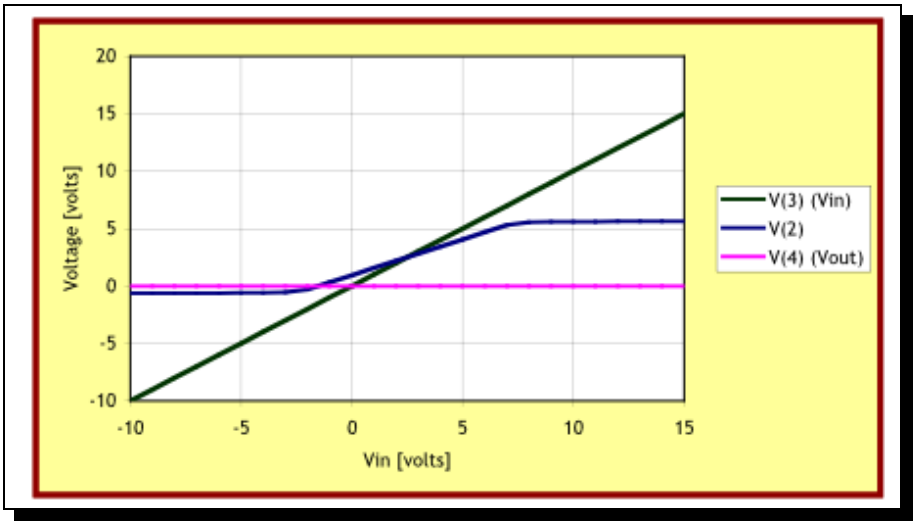

Figure 7.2. DC sweep voltages at Vin, node 2 and Vout.

8. Transient Analysis

Chapter Overview

This chapter illustrates how to set up a transient analysis and includes the following sections:

- Section 8.1, *Transient Analysis Overview*
- Section 8.2, *Defining a Time-Dependent Source*
- Section 8.3, *Transient Calculation Time Steps*
- Section 8.4, *Checkpointing and Restarting*

8.1 Transient Analysis Overview

The transient response analysis simulates the response of the circuit from `TIME=0` to a specified time. Throughout a transient analysis, any or all of the independent sources may have time-dependent values.

In **Xyce**, the transient analysis begins by performing its own bias point calculation at the beginning of the run, using the same method as used for DC sweep. This is required to set the initial conditions for the transient solution as the initial values of the sources may differ from the their DC values.

To run a transient simulation, the circuit netlist file must include a `.TRAN` command with the parameters required for the desired transient analysis (see Appendix A). In addition, the netlist must contain one of the following:

- an independent, transient source (see Table 8.1),
- an initial condition on a reactive element, or
- a time-dependent behavioral modeling source (see Chapter 5.3)

8.2 Defining a Time-Dependent (transient) Source

Overview of Source Elements

Source elements, either voltage or current, are entered in the netlist file as described in Appendix A. Table 8.1 list the time-dependent sources available in **Xyce** for either voltage or current. For voltage sources, the name is preceeded by the letter `V` while current sources are preceeded by the letter `I`.

To use one of these time-dependent or transient sources, the user must place the source element line in the netlist and characterize the transient behavior using the appropriate parameters. Each transient source element has a separate set of parameters dependent on its transient behavior. In this way, the user can create analog sources which produce sine wave, square pulse, exponential pulse, single-frequency FM, and piecewise linear waveforms.

Defining Transient Sources

To define a transient source:

Source Element Name	Description
EXP	Exponential Waveform
PULSE	Pulse Waveform
PWL	Piecewise Linear Waveform
SFFM	Frequency-modulated Waveform
SIN	Sinusoidal Waveform

Table 8.1. Summary of time-dependent sources supported by **Xyce**.

- Select one of the supported sources: independent voltage or current source.
- Choose a transient source type from Table 8.1.
- Provide the transient parameters (see Appendix A) to fully define the source.

Below is an example of an independent sinusoidal voltage source in a circuit netlist. It creates a voltage source between nodes 1 and 5 that oscillates sinusoidally between -5V and +5V with a frequency of 50 KHz.

Example: `Vexample 1 5 SIN(-5V 5V 50KHz)`

8.3 Transient Calculation Time Steps

During the simulation, **Xyce** uses a calculation time step that is continuously adjusted for accuracy and efficiency (see [5]). During periods of circuit idleness the calculation time step is increased, and during dynamic portions of the waveform it is decreased. This release of **Xyce** does not allow the user to specify a maximum time step.

The internal calculation time steps used might not be consistent with the output time steps requested by the user. By default **Xyce** outputs solution results at every time step it calculates. If the user selects output timesteps via the `.OUTPUT` statement (see Chapter 9) then **Xyce** will output results for the closest time step that follows the time requested by the user. There is currently no mechanism for forcing **Xyce** to output at precise user-specified times.

8.4 Checkpointing and Restarting

The .OPTIONS RESTART command (in the netlist) is used to control all checkpoint output and restarting. Checkpointing and associated restart can be extremely useful for long simulations. In essence, **Xyce** allows the user to save the state of the simulation during a run (at intervals the user specifies) (*checkpointing*). This checkpoint data can then be read in to *restart* the simulation from any of the saved (*checkpointed*) time points.

Checkpointing Command Format

- .OPTIONS RESTART JOB=<job name> [INITIAL_INTERVAL=<interval> [<t0> <i0> [<t1> <i1>...]]]

  Here, JOB=<job name> identifies the prefix for restart files. The actual restart files will be the job name appended with the current simulation time (e.g. name1e-05 for JOB=name and simulation time 1e-05 seconds). Furthermore, the INITIAL_INTERVAL=<interval> identifies the initial interval time used for restart output. The <tx ix> intervals identify times (tx) at which the output interval (ix) will change. This functionality is identical that described for the .OPTIONS OUTPUT command (see Section 9.1).

- Example - generate checkpoints at every time step (default):

```
.OPTIONS RESTART JOB=checkpt
```

- Example - generate checkpoints every 0.1 μs:

```
.OPTIONS RESTART JOB=checkpt INITIAL_INTERVAL=0.1us
```

- Example - Initial interval of 0.1 μs, at 1 μs in the simulation, change to interval of 0.5 μs, and at 10 μs change to an interval of 0.1 μs:

```
.OPTIONS RESTART JOB=checkpt INITIAL_INTERVAL=0.1us 1us 0.5us
+ 10us 0.1us
```

Restarting Command Format

- .OPTIONS RESTART <FILE=<filename> | JOB=<job name> START_TIME=<time>>
  + [INITIAL_INTERVAL=<interval> [<t0> <i0> [<t1> <i1> ...]]]

To restart from an existing restart file, the file can be specified by using either the FILE=<filename> parameter to explicitly request a file or JOB=<job name> START_TIME=<time> to specify a file prefix and a specific time. The time must exactly match an output file time for the simulator to correctly load the file. To continue generating restart output files, INITIAL_INTERVAL=<interval> and following intervals can be appended to the command in the same format as described above.

- Example - Restart from checkpoint file at 0.133 μs:

```
.OPTIONS RESTART JOB=checkpt START_TIME=0.133us
```

- Example - Restart from checkpoint file at 0.133 μs :

```
.OPTIONS RESTART FILE=checkpt0.000000133
```

- Example - Restart from 0.133 μs and continue checkpointing at 0.1 μs intervals:

```
.OPTIONS RESTART FILE=checkpt0.000000133 JOB=checkpt_again
+ INITIAL_INTERVAL=0.1us
```

9. Results Output and Evaluation Options

Chapter Overview

This chapter illustrates how to output simulation results to data or output files.

- Section 9.1, *Control of Results Output*
- Section 9.2, *Additional Output Options*
- Section 9.3, *Evaluating Solution Results*

9.1 Control of Results Output

Xyce supports only one solution output command, .PRINT. .PRINT is quite flexible, and supports several output formats.

.PRINT Command

The .PRINT command sends the analysis results to an output file. **Xyce** supports several options on the .PRINT line of netlists that control the format of the output. The syntax for the command is as follows:

- .PRINT <analysis type> [options] <output variable(s)>

Example: .PRINT TRAN FILE=Output.prn V(3) V(2) V(4)

Table 9.1 gives the various options currently available to the .PRINT command. For further information, see Appendix A.

9.2 Additional Output Options

.OPTIONS OUTPUT Command

The main purpose of the .OPTIONS OUTPUT command is to provide control of the frequency at which data is written to files specified by .PRINT TRAN commands. This can be especially useful in controlling the size of the results file for simulations which required a large number of time steps. An additional benefit is that reducing the output frequency from the default, which outputs results at every time-step, can improve performance. The format for controlling the output frequency is:

- .OPTIONS OUTPUT INITIAL_INTERVAL=<interval> [<t0> <i0> [<t1> <i1> ...]]

where INITIAL_INTERVAL=<interval> specifies the starting interval time for output and <tx ix> specifies later simulation times (tx) where the output interval will change to (ix).

The following example shows the output being requested (via the netlist .OPTIONS OUTPUT command) every .1μs for the first 10μs, every 1μs for the next 10μs, and every 5μs for the remainder of the simulation:

Example: .OPTIONS OUTPUT INITIAL_INTERVAL=.1us 10us 1us 20us 5us

Option...	Action...
FORMAT=<STD|NOINDEX|PROBE>	Controls the output format. The STD format outputs data in standard columns. The NOINDEX format is the same as the standard format except that the index column is omitted. The PROBE format specifies that the output should be formatted to be compatible with the PSpice Probe plotting utility. The *default* is STD.
FILE=<output filename>	Allows the user to specify the output filename. The *default* is the netlist filename with the characters ".prn" appended (e.g., foo.cir.prn where foo.cir was the input netlist filename.
WIDTH=<print field width>	Allows the user to control the column width for the output data.
PRECISION=<floating point precision>	Controls the number of significant digits past the decimal point.
FILTER=<filter floor value>	Specifies the absolute value below which output variables will be printed as 0.0.
DELIMITER=<TAB|COMMA>	Specifies an alternate delimiter between columns of output in the STD output format.

Table 9.1. .PRINT command options.

Note: **Xyce** will output data at the next time that is greater-than or equal to the current interval time. This means that output might not correspond exactly to the time intervals due to the adaptive time stepping algorithm.

9.3 Evaluating Solution Results

This section describes how to view graphical waveform analysis of the simulation results generated by **Xyce**. You can use the solution output features of **Xyce** in conjunction with graphing tools (e.g., TecPlot, gnuplot, MS Excel, etc.) to analyze graphically the waveform data created by a **Xyce** circuit simulation (see Figure 9.1 below for an example plot using TecPlot, `http://www.amtec.com`). In addition, **Xyce** is able to output `.csd` files which can be read by the PSpice Probe utility to view the results. See the PSpice Users Guide [1] for instructions on using the Probe tool.

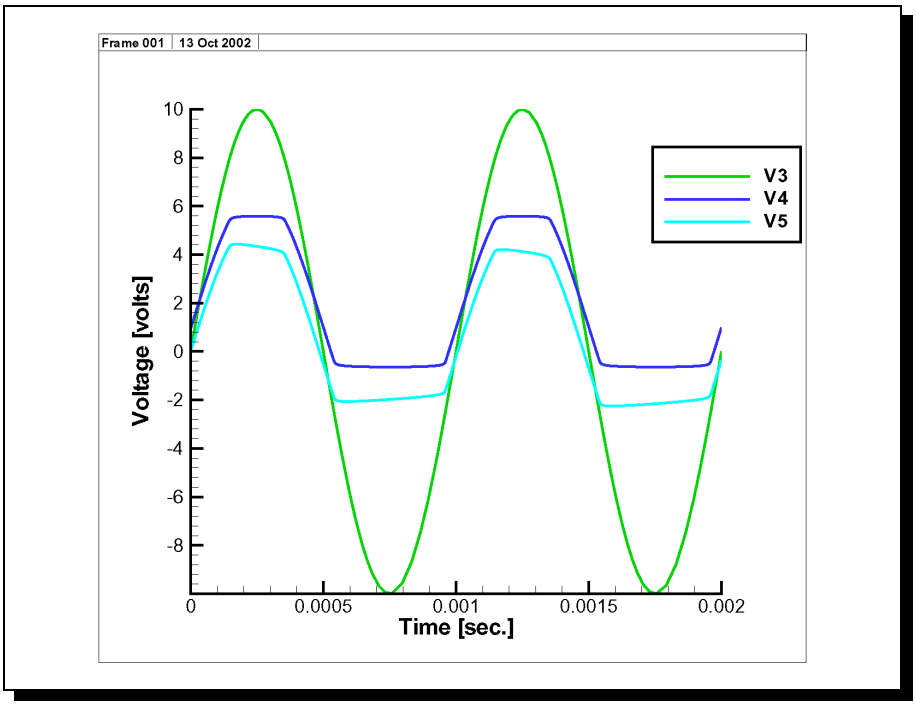

Figure 9.1. TecPlot plot of diode clipper circuit transient response from **Xyce** `.prn` file.

Xyce produces two types of output: the simulation output file and the waveform data file. The calculations and results reported in the simulation output file can be thought of as an audit trail of the simulation. However, graphical analysis of data in the waveform data file is the most useful and accommodating way to evaluate simulation results.

A Netlist Reference

Appendix Overview

This appendix contains reference material directed towards working with circuit analyses in **Xyce** using the netlist interface. Included in this appendix are detailed command descriptions, start-up option definitions and a list of devices supported by the **Xyce** netlist interface.

Netlist Commands

This section outlines the netlist commands that can be used with **Xyce** to setup and control circuit analysis.

Analysis Types

DC Sweep Analysis

Calculates the operating point for the circuit for a range of values for voltage sources. Note that this may be repeated for multiple voltage sources.

General Form

```
.DC <voltage source> <start> <stop> <step>
+ [<voltage source><start> <stop> <step>]...
```

Examples

```
.DC VIN -10 15 1

.DC VDS 0 3.5 0.05 VGS 0 3.5 0.5
```

Transient Analysis

Calculates the time-domain response of a circuit for a specified duration.

General Form

```
.TRAN <print step value> <final time value>
+[<start time value> [<step ceiling value>]]
```

Examples

```
.TRAN 1us 100ms

.TRAN 1ms 100ms 0ms .1ms
```

Arguments and Options

`<print step value>`

Used to calculate the initial time step (see below).

`<final time value>`

Sets the end time (duration) for the analysis.

`[<start time value>]`

Sets the time at which the simulation is to begin. Defaults to zero.

`[<step ceiling value>]`

Sets a maximum time step. Defaults to ((final time value)-(start time value))/10, unless there are breakpoints (see below).

Comments

The transient analysis calculates the circuit's response over an interval of time beginning with `TIME=0` or `TIME=<start time value>` and finishing at `<final time value>`. Use a `.PRINT` (print) statement to get the results of the transient analysis.

Before calculating the transient response **Xyce** computes a bias point for the circuit that is different from the regular bias point. This is necessary because at the start of a transient analysis, the independent sources can have different values than their DC values.

The time integration algorithms within **Xyce** use adaptive time-stepping methods that adjust the time-step size according to the activity in the analysis. The default ceiling for the internal time step is `(<final time value>-<start time value>)/10`. This default ceiling value is automatically adjusted if breakpoints are present, to insure that there are always at least 10 time steps between breakpoints. If the user specifies a ceiling value, however, it overrides any internally generated ceiling values.

Xyce is not strictly compatible with SPICE in its use of the values on the `.TRAN` line. In **Xyce**, `<print step value>` is not used as the printing interval. It is used in determining the initial step size, which is chosen to be the smallest of three quantities: the print step value, the step ceiling value, and 1/200th of the time until the next breakpoint.

Device Modeling

.MODEL (Model Definition)

Modeled device definition.

.SUBCKT (subcircuit)

The .SUBCKT statement begins a subcircuit definition by giving its name, the number and order of its nodes and the names and default parameters that direct its behavior. The .ENDS statement signifies the end of the subcircuit definition.

General Form

```
.SUBCKT <name> [node]*
+ [PARAMS: < <name> = <value> >* ]
...
.ENDS
```

Examples

```
.SUBCKT OPAMP 10 12 111 112 13
...
.ENDS

.SUBCKT FILTER1 INPUT, OUTPUT PARAMS: CENTER=200kHz,
+ BANDWIDTH=20kHz
...
.ENDS

.SUBCKT PLRD IN1 IN2 IN3 OUT1
+ PARAMS: MNTYMXDELY=0 IO_LEVEL=1
...
.ENDS

.SUBCKT 74LS01 A B Y
+ PARAMS: MNTYMXDELY=0 IO_LEVEL=1
...
.ENDS
```

Arguments and Options

`<name>`

The name used to reference a subcircuit.

`[node]*`

An optional list of nodes. This is not mandatory since it is feasible to define a subcircuit without any interface nodes.

`[PARAMS:]`

Keyword that provides values to subcircuits as arguments for use as expressions in the subcircuit.

Comments

A subcircuit designation ends with a `.ENDS` command. The entire netlist between `.SUBCKT` and `.ENDS` is part of the definition. Each time the subcircuit is called via an `X` device, the entire netlist in the subcircuit definition replaces the `X` device.

There must be an equal number of nodes in the subcircuit call and in its definition. As soon as the subcircuit is called, the actual nodes (those in the calling statement) substitute for the argument nodes (those in the defining statement).

Node zero cannot be used in this node list, as it is the global ground node.

Subcircuit references may be nested to any level. However, their definitions cannot be nested. That is, a `.SUBCKT` statement cannot be placed between a `.SUBCKT` and a `.ENDS` statements.

Subcircuits should include only device instantiations and possibly these statements:

- `.IC` (initial bias point condition)
- `.MODEL` (model definition)
- `.PARAM` (parameter)
- `.FUNC` (function)

Models, parameters, and functions defined within a subcircuit are scoped to that definition. That is they are only accessible within the subcircuit definition in which they are included. Further, if a `.MODEL`, `.PARAM` or a `.FUNC` statement is included in the main circuit netlist, it is accessible from the main circuit as well as all subcircuits.

Node, device, and model names are scoped to the subcircuit in which they are defined. It is allowable to use a name in a subcircuit that has been previously used in the main circuit netlist. When the subcircuit is flattened (expanded into the main netlist), all of its names are given a prefix via the subcircuit instance name. For example, `Q17` becomes `X3.Q17` after expansion. After expansion, all names are unique. The single exception occurs in the use of global node names, which are not expanded.

`.ENDS` (end subcircuit)

Marks the end of a subcircuit definition.

Output Control

.PRINT (print)

Send analysis results to an output file. **Xyce** supports several options on the .PRINT line of netlists:

General Form	.PRINT <analysis type> [FORMAT=<STD|NOINDEX|PROBE>] + [FILE=<output filename>] [WIDTH=<print field width>] + [PRECISION=<floating point output precision>] + [FILTER=<absolute value below which a number outputs as 0.0>] + [DELIMITER=<TAB|COMMA>] + <output variable>
Example	.PRINT TRAN FORMAT=PROBE FILE=foobar.csd V(1) .PRINT DC FILE=foobar.txt WIDTH=19 PRECISION=15 FILTER=1.0e-10 + I(VSOURCE5) I(VSOURCE6)

Arguments and Options

`<analysis type>`

Only one analysis type (DC or TRAN) may be given for each `.PRINT` netlist entry.

`[FORMAT=<STD|NOINDEX|PROBE>]`

The output format may be specified using the `FORMAT` option. The `STD` format outputs the data divided up into data columns. The `NOINDEX` format is the same as the `STD` format except that the index column is omitted. The `PROBE` format specifies that the output should be formatted to be compatible with the PSpice Probe plotting utility.

`[FILE=<output filename>]`

Specifies the name of the file to which the output will be written.

`[WIDTH=<print field width>]`

Controls the output width used in formatting the output.

`[PRECISION=<floating point precision >]`

Number of floating point digits past the decimal for output data.

`[FILTER=<filter floor value>]`

Used to specify the absolute value below which output variables will be printed as `0.0`.

`[DELIMITER=<TAB|COMMA>]`

Used to specify an alternate delimiter in the STD or NOINDEX format output.

`<output variable>`

Subsequent to the analysis type and other options is a list of output variables. There is no upper bound on the number of these output variables. The output is divided up into data columns and output according to any specified options (see options given above).

Comments

The values of the output variables are output as a series of columns (one for each output variable).

A netlist may contain multiple `.PRINT` statements.

Netlist Processing

`.END` (End of Circuit)

End of netlist file.

.FUNC (function)

Expression function definition.

.INC or .INCLUDE (include file)

Include specified file in netlist.

The file name can be surrounded by double quotes, "filename", but this is not necessary. The directory for the include file is assumed to be the execution directory unless a full or relative path is given as a part of the file name.

.PARAM (parameter)

Parameter definition.

Miscellaneous Commands

.OPTIONS (Analysis Options)

Set various simulation limits, analysis control parameters and output characters. In general, they use the following format:

```
.OPTIONS <PKG> [<TAG>=<VALUE>]*
```

Exceptions to this format include the OUTPUT and RESTART options that use their own format, which will be defined under their respective descriptions.

The designator *package* refers to the Unified Modeling Language (UML) *package* which refers loosely to a *module* in the code. Thus, the term is used here as identifying a specific *module* to be controlled via *options* set in the netlist input file. The packages which currently support .OPTIONS, and the keywords to use in place of <PKG> are:

Package	PKG keyword
Global:	GLOBAL
Device Model:	DEVICE
Time Integration:	TIMEINT
Nonlinear Solver:	NONLIN
Transient Nonlinear Solver:	NONLIN-TRAN
Linear Solver:	LINSOL
Parallel Distribution:	PARALLEL
Output:	OUTPUT
Restart:	RESTART

As an example, the following netlist line will set the value of **ABSTOL** in the time integration package to 1×10^{-8}:

Example: `.OPTIONS TIMEINT ABSTOL=1E-8`

Below is an outline of the supported packages and their respective options:

Device Model Selections

The device model selections listed in Table A.6 outline the options available for specifying device specific parameters. Some of these (`DEFAS`, `DEFAD`, `TNOM` etc.) have the same meaning as they do for the `.OPTION` line from Berkeley SPICE (3f5).

Device Model (PKG = DEVICE) Tag	Description	Default
DEFAS	MOS Drain Diffusion Area	0.0
DEFAD	MOS Source Diffusion Area	0.0
DEFL	MOS Default Channel Length	1.0E-4
DEFW	MOS Default Channel Width	1.0E-4
GMIN	Minimum Conductance	1.0E-12
TEMP	Temperature	26.85°C (300K)
TNOM	Nominal Temperature	26.85°C (300K)
SCALESRC	Scaling factor for source scaling	0.0
NUMJAC	Numerical Jacobian flag (only use for small problems)	0 (FALSE)
VOLTLIM	Voltage limiting	1 (TRUE)
icFac	This is a multiplicative factor which is applied to right-hand side vector loads of `.IC` initial conditions during the DCOP phase.	10000.0

Device Model (PKG = DEVICE) Tag	Description	Default
LAMBERTW	This flag determines if the lambert-W function should be applied in place of exponentials in hard-to-solve devices. Currently, this capability is implemented in the diode and BJT. Try this for BJT circuits that have convergence problems. For best effect, this option should be tried with voltlim turned off. A detailed explanation of the Lambert-W function, and its application to device modeling can be found in reference [6].	0 (FALSE)
MAXTIMESTEP	Maximum time step size	1.0E+99
DEBUGLEVEL	The higher this number, the more info is output	1
DEBUGMINTIMESTEP	First time-step debug information is output	0
DEBUGMAXTIMESTEP	Last time-step of debug output	65536
DEBUGMINTIME	Same as DEBUGMINTIMESTEP except controlled by time (sec.) instead of step number	0.0
DEBUGMAXTIME	Same as DEBUGMAXTIMESTEP except controlled by time (sec.) instead of step number	100.0

Table A.6: Options for Device Package

Time Integration Options

The time integration options listed in Table A.7 give the available options for helping control the time integration algorithms for transient analysis.

Time Integration (PKG = TIMEINT) Tag	Description	Default
METHOD	Time integration method. This parameter is only relevant when running **Xyce** in transient mode. Supported methods: ■ 1 (Backward Euler) ■ 2 (Backward Differentiation 2) ■ 3 (Trapezoidal)	2 (Backward Differentiation 2)
CONSTSTEP	Constant time step flag. If set to true, the code will use a constant time step in transient mode.	0 (FALSE: use variable time step)
RELTOL	Relative error tolerance	1.0E-04
ABSTOL	Absolute error tolerance	1.0E-08
DOUBLEDCOPSTEP	This option should only be set to TRUE for a PDE device run. PDE devices often have to solve an extra "setup" problem to get the initial condition. This extra setup problem solves a nonlinear poisson equation (see the device appendix for more details), while the normal step solves a full drift-diffusion(DD) problem. The name of this flag refers to the fact that the code is essentially taking two DC operating point steps instead of one. If you set this to TRUE, but have no PDE devices in the circuit, the code will repeat the same identical DCOP step twice. Generally there is no point in doing this.	• 0 (FALSE), if no PDE devices are present. • 1(TRUE) if at least one PDE device is present in the circuit.

Time Integration (PKG = TIMEINT) Tag	Description	Default
FIRSTDCOPSTEP	This is the index of the first DCOP step taken in a simulation for which DOUBLEDCOPSTEP is set to TRUE. The special initialization (nonlinear Poisson) step is referred to as step 0, while the normal (drift-diffusion) step is indexed with a 1. These two options(FIRSTDCOPSTEP and LASTDCOPSTEP) allow you to set the 1st or second DCOP step to be either kind of step. If FIRSTDCOPSTEP and LASTDCOPSTEP are both set to 0, then only the initial setup step happens. If FIRSTDCOPSTEP and LASTDCOPSTEP are both set to 1, then the initialization step doesn't happen, and only the real DD problem is attempted, with a crude initial guess. You should *never* set FIRSTDCOPSTEP to 1 and SECONDDCOPSTEP to 0. Normally, they should always be left as the defaults.	0
LASTDCOPSTEP	This is the second step taken in a simulation for which DOUBLEDCOPSTEP is set to TRUE.	1
DOUBLEDCOPALL	This flag should almost always be false. If it is true, the code will take an extra "setup" step at every step, including transient time steps. The nonlinear poisson equation is thus solved every step to give an initial guess for the current step. This has turned out to not work very well, and may be removed as an option later.	0 (FALSE)

Time Integration (PKG = TIMEINT) Tag	Description	Default
RESETTRANNLS	The nonlinear solver resets its settings for the transient part of the run to something more efficient (basically a simpler set of options with smaller numbers for things like max Newton step). If this is set to false, this resetting is turned off. Normally should be left as default.	1 (TRUE)
BPENABLE	Flag for turning on/off breakpoints (1 = ON, 0 = OFF). It is unlikely anyone would ever set this to FALSE, except to help debug the breakpoint capability.	1 (TRUE:breakpoints are enabled)
EXITTIME	If this is set to nonzero, the code will check the simulation time at the end of each step. If the total time exceeds the exittime, the code will ungracefully exit. This is a debugging option, the point of which is the have the code stop at a certain time during a run without affecting the step size control. If not set by the user, it isn't activated.	-
EXITSTEP	Same as EXITTIME, only applied to step number. The code will exit at the specified step. If not set by the user, it isn't activated.	-

Table A.7: Options for Time Integration Package.

Nonlinear Solver Selections

The nonlinear solver selections listed in Table A.8 provide methods for controlling the nonlinear solver for DC, Transient. Note that the transient parameters for the nonlinear solver operated under the time integrator may be overridden using the Transient Nonlinear Solver Package Options below.

Nonlinear Solver (PKG = NONLIN) Tag	Description	Default
NLSTRATEGY	Nonlinear solution strategy. Supported strategies: ■ 0 (Newton) ■ 1 (Gradient) ■ 2 (Newton-Gradient) ■ 3 (Modified Newton)	0 (Newton)
SEARCHMETHOD	Line-search method used by the nonlinear solver. Supported line-search methods: ■ 0 (Full Newton) ■ 1 (Interval Halving) ■ 2 (Quadratic Linesearch)	0 (Full Newton) (NOTE: for iterative linear solves, the default is Quadratic Linesearch - 2)
ABSTOL	Absolute residual vector tolerance	1.0E-12
RELTOL	Relative residual vector tolerance	1.0E-03
DELTAXTOL	Weighted nonlinear-solution update norm convergence tolerance	1.0
RHSTOL	Residual convergence tolerance (unweighted 2-norm)	1.0E-06
MAXSTEP	Maximum number of Newton steps	200
MAXSEARCHSTEP	Maximum number of line-search steps	9
NORMLVL	Norm level used by the nonlinear solver algorithms *(NOTE: not used for convergence tests)*	2
LINOPT	Linear optimization flag	0 (FALSE)
CONSTRAINTBT	Constraint backtracking flag	0 (FALSE)
CONSTRAINTMAX	Global maximum setting for constraint backtracking	DBL_MAX (Machine Dependent Constant)

Nonlinear Solver (PKG = NONLIN) Tag	Description	Default
CONSTRAINTMIN	Global minimum setting for constraint backtracking	-DBL_MAX (Machine Dependent Constant)
CONSTRAINTCHANGE	Global percentage-change setting for constraint backtracking	sqrt(DBL_MAX) (Machine Dependent Constant)
IN_FORCING	Inexact Newton-Krylov forcing flag	1 (TRUE)
DIRECTLINSOLV	Direct (vs. iterative) linear solution flag *(Serial Only!)*	0 (FALSE)
DLSDEBUG	Debug output for direct linear solver	0 (FALSE)
DEBUGLEVEL	The higher this number, the more info is output	1
DEBUGMINTIMESTEP	First time-step debug information is output	0
DEBUGMAXTIMESTEP	Last time-step of debug output	99999999
DEBUGMINTIME	Same as DEBUGMINTIMESTEP except controlled by time (sec.) instead of step number	0.0
DEBUGMAXTIME	Same as DEBUGMAXTIMESTEP except controlled by time (sec.) instead of step number	1.0E+99

Table A.8: Options for Nonlinear Solver Package.

Transient Nonlinear Solver Package Options

The nonlinear solver selections listed in Table A.9 provide methods for controlling the nonlinear solver for Transient analysis. These override the defaults and any that were set simply in the Nonlinear Solver Package (e.g., set MAXSTEP=5 for transient).

Nonlinear Solver (PKG = NONLIN-TRAN) Tag	Description	Default
NLSTRATEGY	Nonlinear solution strategy. Supported strategies: ■ 0 (Newton) ■ 1 (Gradient) ■ 2 (Newton-Gradient) ■ 3 (Modified Newton)	0 (Newton)

Nonlinear Solver (PKG = NONLIN-TRAN) Tag	Description	Default
SEARCHMETHOD	Line-search method used by the nonlinear solver. Supported line-search methods: ■ 0 (Full Newton) ■ 1 (Interval Halving) ■ 2 (Quadratic Linesearch)	0 (Full Newton)
ABSTOL	Absolute residual vector tolerance	1.0E-06
RELTOL	Relative residual vector tolerance	1.0E-03
DELTAXTOL	Weighted nonlinear-solution update norm convergence tolerance	0.33
RHSTOL	Residual convergence tolerance (unweighted 2-norm)	10.0 * ABSTOL
MAXSTEP	Maximum number of Newton steps	40
MAXSEARCHSTEP	Maximum number of line-search steps	9
NORMLVL	Norm level used by the nonlinear solver algorithms *(NOTE: not used for convergence tests)*	2
LINOPT	Linear optimization flag	0 (FALSE)
CONSTRAINTBT	Constraint backtracking flag	0 (FALSE)
CONSTRAINTMAX	Global maximum setting for constraint backtracking	DBL_MAX (Machine Dependent Constant)
CONSTRAINTMIN	Global minimum setting for constraint backtracking	-DBL_MAX (Machine Dependent Constant)
CONSTRAINTCHANGE	Global percentage-change setting for constraint backtracking	sqrt(DBL_MAX) (Machine Dependent Constant)
IN_FORCING	Inexact Newton-Krylov forcing flag	1 (TRUE)
DIRECTLINSOLV	Direct (vs. iterative) linear solution flag *(Serial Only!)*	0 (FALSE)
DLSDEBUG	Debug output for direct linear solver	0 (FALSE)
DEBUGLEVEL	The higher this number, the more info is output	1
DEBUGMINTIMESTEP	First time-step debug information is output	0
DEBUGMAXTIMESTEP	Last time-step of debug output	99999999

Nonlinear Solver (PKG = NONLIN-TRAN) Tag	Description	Default
DEBUGMINTIME	Same as DEBUGMINTIMESTEP except controlled by time (sec.) instead of step number	0.0
DEBUGMAXTIME	Same as DEBUGMAXTIMESTEP except controlled by time (sec.) instead of step number	1.0E+99

Table A.9: Options for Nonlinear Solver Package under transient operation.

Linear Solver Options

Xyce uses both a sparse-direct solver as well as Krylov iterative methods for the solution of the linear equations generated by Newton's method. For the advanced users, there are a variety of options that can be set to help improve these solvers. Many of these options (for the Krylov solvers) are simply passed through to the underlying Trilinos/Aztec solution settings and thus have an "AZ_" prefix on the flag; the "AZ_" options are all case-sensitive. The list in Table A.10 only provides a partial list of the more commonly used Trilinos/Aztec options. For a full list of the available options, please see the Aztec User's Guide [7] available for download at http://www.cs.sandia.gov/CRF/Aztec1. However, for most users, the default options should prove adequate.

Linear Solver (PKG = LINSOL) Tag	Description	Default
AZ_tolerance	Iterative solver convergence tolerance	1.0E-10
AZ_max_iter	Maximum number of iterative solver iterations	100
AZ_precond	Iterative solver preconditioner flag	AZ_dom_decomp
AZ_subdomain_solve	Subdomain solution for domain decomposition preconditioners	AZ_icc
AZ_ilut_fill	Approximate allowed fill-in factor for the ILUT preconditioner	3.0
AZ_drop	Specifies drop tolerance used in conjunction with LU or ILUT preconditioners	1.0E-03
TR_loadbalance	Perform load-balance on the linear system	0 (FALSE)

Linear Solver (PKG = LINSOL) Tag	Description	Default
TR_filter	Providing a non-zero value for THRESHOLD triggers filtering of the Jacobian entries by ROW_SUM * THRESHOLD	0

Table A.10: Options for Linear Solver Package.

Parallel Options

These options are used to control partitioning of the problem in distributed memory parallel environments. There are currently only 2 parameters available.

Parallel (PKG = PARALLEL) Tag	Description	Default
Partitioner	Choice of partitioners	0 (Chaco)
DistribIndSrcNodes	Cutoff threshold for Independent Source connectivity before distibution of source	1.0

Table A.11: Options for Parallel Support.

Output Options

The main purpose of the .OPTIONS OUTPUT command is to allow control of the output frequency of data to files specified by .PRINT TRAN commands. The format is:

```
.OPTIONS OUTPUT INITIAL_INTERVAL=<interval> [<t0> <i0> [<t1> <i1>...]]
```

where INITIAL_INTERVAL=<interval> specifies the starting interval time for output and <tx> <ix> specifies later simulation times <tx> where the output interval will change to <ix>.

Note: Xyce will output data at the next time that is greater-than or equal to the current interval time. This means that output will not exactly correspond to the time intervals due to the adaptive time stepping algorithms.

Checkpointing and Restarting Options

The .OPTIONS RESTART command is used to control all checkpoint output and restarting.

- Checkpointing command format:

```
.OPTIONS RESTART JOB=<job prefix>
+ [INITIAL_INTERVAL=<initial interval time>
+  [<t0> <i0> [<t1> <i1>...]]]
```

where JOB=<job prefix> identifies the prefix for restart files. The actual restart files will be the job name with the current simulation time appended (e.g. name1e-05 for JOB=name and simulation time 1e-05 seconds). Furthermore, INITIAL_INTERVAL=<initial interval time> identifies the initial interval time used for restart output. The <tx> <ix> intervals identify times <tx> at which the output interval (ix) should change. This functionality is identical that described for the .OPTIONS OUTPUT command.

To generate checkpoints at every time step (default):

Example: .OPTIONS RESTART JOB=checkpt

To generate checkpoints every 0.1 μs:

Example: .OPTIONS RESTART JOB=checkpt INITIAL_INTERVAL=0.1us

To specify an initial interval of 0.1 μs, at 1 μs change to interval of 0.5 μs, and at 10 μs change to interval of 0.1 μs:

Example:

```
.OPTIONS RESTART JOB=checkpt INITIAL_INTERVAL=0.1us 1.0us
+ 0.5us 10us 0.1us
```

To restart from an existing restart file, specify the file by either FILE=<restart file name> to explicitly use a restart file or by JOB=<job name> START_TIME=<specified name> to specify a file prefix and a specified time. The time must exactly match an output file time for the simulator to correctly identify the correct file. To continue generating restart output files, INITIAL_INTERVAL=<interval> and following intervals can be appended to the command in the same format as described above. Here are several examples:

- Restarting command format:

```
.OPTIONS RESTART <FILE=<restart file name> |
+ JOB=<job name> START_TIME=<time>)>
+ [ INITIAL_INTERVAL=<interval> [<t0> <i0> [<t1> <i1> ...]]]
```

Example restarting from checkpoint file at 0.133 μs:

Example: .OPTIONS RESTART JOB=checkpt START_TIME=0.133us

To restart from checkpoint file at 0.133 μs:

Example: `.OPTIONS RESTART FILE=checkpt0.000000133`

Restarting from 0.133 μs and continue checkpointing at 0.1 μs intervals:

Example:

```
 .OPTIONS RESTART FILE=checkpt0.000000133 JOB=checkpt_again
+ INITIAL_INTERVAL=0.1us
```

* (Comment)

Create a netlist comment line.

; (In-line Comment)

Add a netlist in-line comment

+ (Line Continuation)

Continue the text of the previous line

Analog Devices

Xyce supports many analog devices, including sources, subcircuits and behavioral models. This appendix serves as a reference for the analog devices supported by **Xyce**. Each device is described separately and includes the following information, if applicable:

- a description and an example of the correct netlist syntax.
- the matching model types and their description.
- the matching list of model parameters and associated descriptions.
- the corresponding and characteristic equations for the model (as required).
- references to publications on which the model is based.

You can also create models and macromodels using the `.MODEL` (model definition) and `.SUBCKT` (subcircuit) statements, respectively.

Please note that the characteristic equations are provided to give a general representation of the device behavior. The actual **Xyce** implementation of the device may be slightly different in order to improve, for example, the robustness of the device.

Table 4.2 gives a summary of the analog device types and the form of their netlist formats. Each of these in described below in detail.

Capacitor

General Form	C<name> <(+) node> <(-) node> [model name] <value> [IC=<initial value>][L=<length> W=<width>]
Age-aware Form	C<name> <(+) node> <(-) node> <value> [D=<coeff>][AGE=<age> (hours)]
Examples	CM12 2 4 5.288e-13 CLOAD 1 0 4.540pF IC=1.5V CFEEDBACK 2 0 CMOD 1.0pF CAGED 2 3 4.0uF D=0.0233 AGE=86200
Model Form	.MODEL <model name> C [model parameters]
Parameters and Options	(+) and (-) nodes Polarity definition for a positive voltage across the capacitor. The first node is defined as positive. Therefore, the voltage across the component is the first node voltage minus the second node voltage. [model name] If [model name] is omitted, then <value> is the capacitance in farads. If [model name] is given then the value is determined from the model parameters; see the capacitor value formula below. <value> The capacitance value in farads. <initial value> The initial voltage across the capacitor during the bias point calculation. <coeff> Degradation coefficient for age-aware capacitor model. <age> Capacitor age in hours. <length> Geometry length for semiconductor capacitor. <width> Geometry width for semiconductor capacitor.

Comments	Positive current flows through the capacitor from the `(+)` node to the `(-)` node. In general, capacitors should have a positive capacitance value (`<value>` property). In all cases, the capacitance must not be zero. However, cases exist when a negative capacitance value may be used. This occurs most often in filter designs that analyze an RLC circuit equivalent to a real circuit. When transforming from the real to the RLC equivalent, the result may contain a negative capacitance value. In a transient run, negative capacitance values may cause the simulation to fail due to instabilities they cause in the time integration algorithms. The age-aware capacitor can only be used without a model.

Model Parameters

Table A.13 gives the available model parameters for the capacitor.

Model parameters	Description	Units	Default
C	Capacitance Multiplier		1.0
VC1	Linear Voltage Coefficient	$volt^{-1}$	0.0
VC2	Quadratic Voltage Coefficient	$volt^{-2}$	0.0
TC1	Linear Temperature Coefficient	$°C^{-1}$	0.0
TC2	Quadratic Temperature Coefficient	$°C^{-2}$	0.0
TNOM	Parameter Measurement Temperature	°C	27.0
D	Degradation Coefficient	-	0.0233
AGE	Capacitor model age	hours	0.0
L	Semiconductor model length	m	1.0
W	Semiconductor model width	m	1.0E-6
CJ	Junction bottom capacitance	F/m^2	0.0
CJSW	Junction sidewall capacitance	F/m	0.0
NARROW	Narrowing due to side etching	m	0.0

Table A.13: Capacitor Model Parameters.

Capacitor Equations

Capacitance Value Formula

If `[model name]` is specified, then the value is given by:

$$< \texttt{value} > \cdot C(1 + \mathbf{VC1} \cdot V + \mathbf{VC2} \cdot V^2)(1 + \mathbf{TC1} \cdot (T - T_0) + \mathbf{TC2} \cdot (T - T_0)^2)$$

where <value> is normally positive (though it can be negative, but not zero). T_0 is the nominal temperature (set using $\mathbf{TNOM}$ option).

Age-aware Formula

If $\mathbf{AGE}$ is given, then the capacitance is:

$$C[1 - D\log(\mathbf{AGE})]$$

Semiconductor Formula

If $\mathbf{L}$ and $\mathbf{W}$ are given, then the capacitance is:

$$\mathbf{CJ}(\mathbf{L} - \mathbf{NARROW})(\mathbf{W} - \mathbf{NARROW}) + 2 \cdot \mathbf{CJSW}(\mathbf{L} - \mathbf{W} + 2 \cdot \mathbf{NARROW})$$

Capacitor Noise Equation

There is no noise model for the capacitor.

Inductor

General Form	`L<name> <(+) node> <(-) node> [model name] <value> + [IC=<initial value>]`

Examples	`L1 1 5 3.718e-08`
	`LLOAD 3 6 4.540mH IC=2mA`
	`LSENSE 2 0 LMOD 0.3`

Model Form	`.MODEL <model name> I [model parameters]`

Parameters and Options	`(+)` and `(-)` nodes
	Polarity definition for a positive voltage across the inductor. The first node is defined as positive. Therefore, the voltage across the component is the first node voltage minus the second node voltage.
	`[model name]`
	If `[model name]` is omitted, then `<value>` is the inductance in Henrys. If `[model name]` is given then the value is determined from the model parameters; see the inductance value formula below. If the inductor is associated with a core model, then the effective value is the number of turns on the core. Otherwise, the effective value is the inductance. See the Model Form statement for the K device in Inductor coupling (and magnetic core) for more information on the core model.
	`<initial value>`
	The initial current through the inductor during the bias point calculation.

Comments	In general, inductors should have a positive inductance value (`VALUE` property). In all cases, the inductance must not zero. However, cases exist when a negative value may be used. This occurs most often in filter designs that analyze an RLC circuit equivalent to a real circuit. When transforming from the real to the RLC equivalent, the result may contain a negative inductance value.

Model Parameters

Table A.15 gives the available model parameters for the inductor.

Model parameters	Description	Units	Default
`L`	Inductance Multiplier		1.0
`IL1`	Linear Current Coefficient	amp^{-1}	0.0

Model parameters	Description	Units	Default
IL2	Quadratic Current Coefficient	amp^{-2}	0.0
TC1	Linear Temperature Coefficient	$^{\circ}\mathrm{C}^{-1}$	0.0
TC2	Quadratic Temperature Coefficient	$^{\circ}\mathrm{C}^{-2}$	0.0
TNOM	Parameter Measurement Temperature	$^{\circ}\mathrm{C}$	27.0

Table A.15: Inductor Model Parameters.

Inductor Equations

Inductance Value Formula

If [model name] is specified, then the value is given by:

$$< \texttt{value} > \cdot L \cdot (1 + \mathbf{IL1} \cdot I + \mathbf{IL2} \cdot I^2)(1 + \mathbf{TC1} \cdot (T - T_0) + \mathbf{TC2} \cdot (T - T_0)^2)$$

where <value> is normally positive (though it can be negative, but not zero). T_0 is the nominal temperature (set using $\mathbf{TNOM}$ option).

Inductor Noise Equation

There is no noise model for the inductor.

Mutual Inductors

General Form

```
K<name> L<inductor name> [L<inductor name>*] <coupling value>
+ [model name]
```

Examples

```
KTUNED L3OUT  L4IN .8
KTRNSFRM LPRIMARY LSECNDRY 1
KXFRM L1 L2  L3  L4 .98 KPOT_3C8
```

Model Form

```
.MODEL <model name> CORE [model parameters]
```

Parameters and Options

`L<inductor name> [L<inductor name>*]`

Identifies the inductors to be coupled. The inductors are coupled and in the dot notation the dot is placed on the first node of each inductor. The polarity is determined by the order of the nodes in the L devices and not by the order of the inductors in the K statement.

`<coupling value>`

The coefficient of mutual coupling, which must be between -1.0 and 1.0.

This coefficient is defined by the equation

<coupling value> = $\frac{M_{ij}}{L_i L_j}$

where

L_i is the inductance of the ith named inductor in the K-line

M_{ij} is the mutual inductance between L_i and L_j

For transformers of normal geometry, use 1.0 as the value. Values less than 1.0 occur in air core transformers when the coils do not completely overlap.

`<model name>`

If `<model name>` is present, four things change:

- The mutual coupling inductor becomes a nonlinear, magnetic core device.
- The inductors become windings, so the number specifying inductance now specifies the number of turns.
- The list of coupled inductors could be just one inductor.
- A model statement is required to specify the model parameters.

Model Parameters

Table A.17 gives the available model parameters for mutual inductors.

Model parameters	Description	Units	Default
AREA	mean magnetic cross-sectional area	cm^2	0.01
GAP	effective air gap length	cm	0.0
PATH	total mean magnetic path length	cm	1.0
MS	saturation magnetization	$\mathrm{amp/m}$	1.0e6
A	thermal energy parameter	$\mathrm{amp/m}$	1000.0
C	domain flexing constant		0.2
ALPHA	domain coupling parameter		5.0e-5
KIRR	irreversible magnetization	$\mathrm{amp/m}$	500.0
DELV	modeling parameter	V	0.001
BETAH	modeling parameter		0.00001
BETAM	modeling parameter		3.124e-5
K	alternative specification of KIRR for compatibility with PSpice	$\frac{A}{m}$	500.0
PACK	PSpice compatibility parameter, ignored by **Xyce**		
LEVEL	PSpice compatibility parameter, ignored by **Xyce**		

Table A.17: Inductor Model Parameters.

Resistor

General Form	`R<name> <(+) node> <(-) node> [model name] <value>` `+ [L=<length>] [W=<width>]`

Examples	`R1 1 2 2K`
	`RLOAD 3 6 4.540 TC=.01,-.001`
	`RFEEDBACK 2 0 RMOD 1K`

Model Form	`.MODEL <model name> RES [model parameters]`

Parameters and Options	`(+)` and `(-)` nodes
	Polarity definition for a positive voltage across the resistor. The first node is defined as positive. Therefore, the voltage across the component is the first node voltage minus the second node voltage. Positive current flows from the positive node (first node) to the negative node (second node).
	`[model name]`
	If `[model name]` is omitted, then `<value>` is the resistance in Ohms. If `[model name]` is given then the value is determined from the model parameters; see the resistance value formula below.
	`<length>` and `<width>`
	Length and width for semiconductor resistance model

Comments	Resistors must have a positive resistance value (`<value>` property) - the resistance must not be zero.

Model Parameters

Table A.19 gives the available model parameters for the resistor.

Model parameters	Description	Units	Default
R	Resistance Multiplier		1.0
TC1	Linear Temperature Coefficient	$^{\circ}C^{-1}$	0.0
TC2	Quadratic Temperature Coefficient	$^{\circ}C^{-2}$	0.0
TCE	Exponential Temperature Coefficient	$\%/^{\circ}C$	0.0
TNOM	Parameter Measurement Temperature	$^{\circ}C$	27.0
RSH	Sheet Resistance	ohm	0.0
NARROW	Narrowing due to side etching	m	0.0

Table A.19: Resistor Model Parameters.

Resistor Equations

Resistance Value Formula

If [model name] is included *and* $\mathbf{TCE}$ is given, then the resistance is:

$$< \texttt{value} > \cdot R \cdot 1.01^{\mathbf{TCE}(T-T_0)}$$

where <value> is usually positive although it may be negative but never zero. T_0 is the nominal temperature (set using $\mathbf{TNOM}$ option).

If [model name] is included *and* $\mathbf{TCE}$ is not given, then the resistance is:

$$< \texttt{value} > \cdot R(1 + \mathbf{TC1} \cdot (T - T_0) + \mathbf{TC2} \cdot (T - T_0)^2)$$

where <value> is typically positive although it may be negative but never zero.

If $\mathbf{L}$ and $\mathbf{W}$ are given, the resistance is:

$$\mathbf{RSH}\frac{[\mathbf{L} - \mathbf{NARROW}]}{[\mathbf{W} - \mathbf{NARROW}]}$$

Resistor Noise Equation

Resistor noise is calculated assuming a 1.0-hertz bandwidth. The resistor generates thermal noise using the following spectral power density (per unit bandwidth):

$$i^2 = \frac{4kT}{R}$$

Diode

General Form	D<name> <(+) node> <(-) node> <model name> [area value]
Examples	DCLAMP 1 0 DMOD D2 15 17 SWITCH 1.5
Model Form	.MODEL <model name> D [model parameters]
Parameters and Options	<(+) node> The anode. <(-) node> The cathode. [area value] Scales IS, ISR, IKF, RS, CJO, and IBV, and has a default value of 1. IBV and BV are both specified as positive values.
Comments	The diode is modeled as an ohmic resistance (RS/area) in series with an intrinsic diode. Positive current is current flowing from the anode through the diode to the cathode.

Diode Operating Temperature

Model parameters can be assigned unique measurement temperatures using the TNOM model parameter.

Diode level selection

Two distinct implementations of the diode are available. These are selected by using the LEVEL model parameter. The default implementation is based on SPICE 3F5, and may be explicitly specified using LEVEL=1 in the model parameters, but is also selected if no LEVEL parameter is specified. The second implementation is based on the AIM-SPICE diode model[8] as modified to include radiation effects[9]. This second implementation is selected using the LEVEL=3 model parameter. There is no LEVEL=2 model in this version of **Xyce**.

Model Parameters

Table A.21 gives the available model parameters for the LEVEL=1 diode.

Model parameters	Description	Units	Default
BV	Reverse Breakdown Knee Voltage	volt	1E99
CJO	Zero-bias p-n Depletion Capacitance	farad	0.0

Model parameters	Description	Units	Default
EG	Activation Energy	eV	1.11
FC	Forward-bias Depletion Capacitance Coefficient	-	0.5
IBV	Reverse Breakdown Knee Current	amp	1E-3
IS	Saturation Current	amp	1E-14
M	p-n Grading Parameter	-	0.5
N	Emission Coefficient	-	1.0
RS	Parasitic Resistance	ohm	0.0
TT	Transit Time	sec	0.0
TNOM	Parameter Measurement Temperature	°C	27.0
VJ	p-n Potential	volt	1.0
XTI	**IS** Temperature Exponent		3.0

Table A.21: Diode Model Parameters.

Table A.22 gives the available model parameters for the LEVEL=3 diode. In all cases, voltage parameters related to the "radiation voltage" are relative to the radiation generation rate (e.g. if PULSEV2 is 4.3 and GRORDER is 26, the radiation dose at the top of the pulse is $4.3 \times 10^{26} m^{-3} s^{-1}$, or 1×10^{7}rads^{-s}).

Model parameters	Description	Units	Default
KF	Flicker Noise Coefficient	-	0.0
AF	Flicker Noise Exponent	-	1.0
BV	Reverse Breakdown Knee Voltage	volt	1E99
CJO	Zero-bias p-n Depletion Capacitance	farad	0.0
EG	Activation Energy	eV	1.11
FC	Forward-bias Depletion Capacitance Coefficient	-	0.5
IBV	Reverse Breakdown Knee Current	amp	1E-3
IS	Saturation Current	amp	1E-14
M	p-n Grading Parameter	-	0.5
N	Emission Coefficient	-	1.0
RS	Parasitic Resistance	ohm	0.0
TT	Transit Time	sec	0.0
TNOM	Parameter Measurement Temperature	°C	27.0
VJ	p-n Potential	volt	1.0
XTI	**IS** Temperature Exponent	-	3.0

Model parameters	Description	Units	Default
IKF	High-injection knee current	amp	0.0
ISR	Recombination current	amp	0.0
NR	Emission coefficient for ISR	-	2.0
ISNPP	Is this an n-p-p+ diode?	boolean	0 (false)
ISPNN	Is this an p-n-n+ diode?	boolean	0 (false)
FUNCTIONTYPE	Selects the shape of the radiation pulse	-	0 (none)
GRORDER	Order of magnitude of radiation generation rate	-	25
PERMITTIVITY	Dielectric permittivity of the semiconductor	F/m	1.0443E-10
NA	Doping density in p- region	m^{-3}	5E20
NAHI	Doping density in p+ region	m^{-3}	1E24
ND	Doping density in n- region	m^{-3}	3E25
NDHI	Doping density in n+ region	m^{-3}	3E25
NI	Intrinsic concentration	m^{-3}	1.45E16
WN	Width of the n- region	m	2E-6
WNHI	Width of the n+ region	m	1E-6
WP	Width of the n- region	m	5E-5
WPHI	Width of the n+ region	m	1E-5
PULSEV1	Initial value of radiation pulse (FUNCTIONTYPE=1)	volt	0.0
PULSEV2	Maximum value of radiation pulse (FUNCTIONTYPE=1)	volt	4.3
PULSETD	Delay of radiation pulse (FUNCTIONTYPE=1)	sec	0.0
PULSETR	Rise time of radiation pulse (FUNCTIONTYPE=1)	sec	1E-5
PULSETF	Fall time of radiation pulse (FUNCTIONTYPE=1)	sec	1E-5
PULSEPW	Width of radiation pulse (FUNCTIONTYPE=1)	sec	1E-5
PULSEPER	Period of radiation pulse (FUNCTIONTYPE=1)	sec	1E10
SINVO	Initial value of sinusoidal radiation voltage (FUNCTIONTYPE=2)	volt	0.0
SINVA	Amplitude of sinusoidal radiation voltage (FUNCTIONTYPE=2)	volt	0.0

Model parameters	Description	Units	Default
SINFREQ	Frequency of sinusoidal radiation voltage (FUNCTIONTYPE=2)	Hz	1.0
SINTD	Delay of sinusoidal radiation voltage (FUNCTIONTYPE=2)	sec	0.0
SINTHETA	Inverse of time constant for exponential envelope of sinusoidal radiation voltage (FUNCTIONTYPE=2)	sec^{-1}	0.0
EXPV1	Initial value of exponential cusp radiation voltage (FUNCTIONTYPE=3)	volt	0.0
EXPV2	Height of exponential cusp radiation voltage (FUNCTIONTYPE=3)	volt	0.0
EXPTD1	Rise delay time of exponential cusp radiation voltage (FUNCTIONTYPE=3)	sec	0.0
EXPTAU1	Rise time constant of exponential cusp radiation voltage (FUNCTIONTYPE=3)	sec	1e-2
EXPTD2	Fall delay time of exponential cusp radiation voltage (FUNCTIONTYPE=3)	sec	1E10
EXPTAU2	Fall time constant of exponential cusp radiation voltage (FUNCTIONTYPE=3)	sec	1E10
SFFMVO	Offset of single frequency FM curve (FUNCTIONTYPE=4)	volt	0.0
SFFMVA	Amplitude of single frequency FM curve (FUNCTIONTYPE=4)	volt	0.0
SFFMFC	Carrier frequency of single frequency FM curve (FUNCTIONTYPE=4)	Hz	1.0
SFFMMDI	Modulation index of single frequency FM curve (FUNCTIONTYPE=4)	-	0.0
SFFMFS	Signal frequency of single frequency FM curve (FUNCTIONTYPE=4)	Hz	0.0
PWLT1	Time of point 1 of piecewise linear curve (FUNCTIONTYPE=5)	sec	1
PWLT2	Time of point 2 of piecewise linear curve (FUNCTIONTYPE=5)	sec	2
PWLT3	Time of point 3 of piecewise linear curve (FUNCTIONTYPE=5)	sec	3
PWLT4	Time of point 4 of piecewise linear curve (FUNCTIONTYPE=5)	sec	4
PWLT5	Time of point 5 of piecewise linear curve (FUNCTIONTYPE=5)	sec	5
PWLV1	Height of point 1 of piecewise linear curve (FUNCTIONTYPE=5)	volt	0

Model parameters	Description	Units	Default
PWLV2	Height of point 2 of piecewise linear curve (FUNCTIONTYPE=5)	volt	0
PWLV3	Height of point 3 of piecewise linear curve (FUNCTIONTYPE=5)	volt	0
PWLV4	Height of point 4 of piecewise linear curve (FUNCTIONTYPE=5)	volt	0
PWLV5	Height of point 5 of piecewise linear curve (FUNCTIONTYPE=5)	volt	0
LINET1	Time of point 1 for linear radiation voltage (FUNCTIONTYPE=6)	sec	0
LINEV1	Height of point 1 for linear radiation voltage (FUNCTIONTYPE=6)	volt	0
LINEK	Slope linear radiation voltage (FUNCTIONTYPE=6)	-	0
RAUGN	Auger recombination rate for electrons in p-region	cm^6/s	1.1E-42
RAUGP	Auger recombination rate for holes in n-region	cm^6/s	3E-43
NDN	Diffusion constant of electrons in n-region at low injection	m^2/s	1E-3
NDP	Diffusion constant of holes in n-region at low injection	m^2/s	7E-4
TAUP0	Hole lifetime in n-region at low injection	sec	2e-8
TAUINFP	Hole lifetime in n-region at saturated SRH condition	sec	4E-8
PDN	Diffusion constant of electrons in p-region at low injection	m^2/s	4E-3
PDP	Diffusion constant of holes in p-region at low injection	m^2/s	5E-4
TAUN0	Electron lifetime in p-region at low injection	sec	1e-6
TAUINFN	Electron lifetime in p-region at saturated SRH condition	sec	2e-6
PDNHI	Diffusion constant of electrons in p+-region at low injection	m^2/s	5E-3
PDPHI	Diffusion constant of holes in p+-region at low injection	m^2/s	1E-4
TAUNOHI	Electron lifetime in p+-region at low injection	sec	1e-7
TAUINFNHI	Electron lifetime in p+-region at saturated SRH condition	sec	2E-7

Model parameters	Description	Units	Default
NDNHI	Diffusion constant of electrons in n+-region at low injection	m^2/s	1e-3
NDPHI	Diffusion constant of holes in n+-region at low injection	m^2/s	2e-4
TAUPOHI	Hole lifetime in n+-region at low injection	sec	2e-8
TAUINFPHI	Hole lifetime in n+-region at saturated SRH condition	sec	4e-8

Table A.22: Diode Level 3 Model Parameters.

Diode Equations

The equations in this section use the following variables:

$$\begin{aligned} V_{di} &= \text{voltage across the intrinsic diode only} \\ V_{th} &= k \cdot T/q \text{ (thermal voltage)} \\ k &= \text{Boltzmann's constant} \\ q &= \text{electron charge} \\ T &= \text{analysis temperature (Kelvin)} \\ T_0 &= \text{nominal temperature (set using } \mathbf{TNOM} \text{ option)} \\ \omega &= \text{Frequency (Hz)} \end{aligned}$$

Other variables are listed above in the diode model parameters.

Level=1

The level 1 diode is based on the Spice3f5 level 1 model.

DC Current (Level=1)

The intrinsic diode current consists of forward and reverse bias regions where

$$I_D = \begin{cases} \mathbf{IS} \cdot \left[\exp\left(\frac{V_{di}}{\mathbf{N}V_{th}}\right) - 1\right], & V_{di} > -3.0 \cdot \mathbf{N}V_{th} \\ -\mathbf{IS} \cdot \left[1.0 + \left(\frac{3.0 \cdot \mathbf{N}V_{th}}{V_{di} \cdot q}\right)^3\right], & V_{di} < -3.0 \cdot \mathbf{N}V_{th} \end{cases}$$

When $\mathbf{BV}$ is explicitly given in the model statement, an exponential current model is substituted when $V_{di} < -\mathbf{BV}$.

This equation for I_D is

$$I_D = -\mathbf{IS} \cdot \exp\left(-\frac{BV + V_{di}}{\mathbf{N}V_{th}}\right)$$

Note here that if $\mathbf{IBV}$ is set by the user, then $\mathbf{BV}$ is changed internally to ensure consistency. This is the equation actually used by **Xyce** but the value of $\mathbf{BV}$ is not required to be that specified by the user.

Capacitance (Level=1)

The p-n diode capacitance consists of a depletion layer capacitance C_d and a diffusion capacitance C_{dif}. The first is given by

$$C_d = \begin{cases} \mathbf{CJ} \cdot \mathbf{AREA} \left(1 - \frac{V_{di}}{\mathbf{VJ}}\right)^{-\mathbf{M}}, & V_{di} \leq \mathbf{FC} \cdot \mathbf{VJ} \\ \frac{\mathbf{CJ} \cdot \mathbf{AREA}}{\mathbf{F2}} \left(\mathbf{F3} + \mathbf{M}\frac{V_{di}}{\mathbf{VJ}}\right), & V_{di} > \mathbf{FC} \cdot \mathbf{VJ} \end{cases}$$

The diffusion capacitance (sometimes referred to as the transit time capacitance) is

$$C_d if = \mathbf{TT} G_d = \mathbf{TT} \frac{dI_D}{dV_{di}}$$

where G_d is the junction conductance.

Temperature Effects (Level=1)

The diode model contains explicit temperature dependencies in the ideal diode current, the generation/recombination current and the breakdown current. Further temperature dependencies are present in the diode model via the saturation current I_S, the depletion layer junction capacitance CJ, the junction potential V_J.

$$\begin{aligned} V_t(T) &= \frac{kT}{q} \\ V_{tnom}(T) &= \frac{k\mathbf{TNOM}}{q} \\ E_g(T) &= E_{g0} - \frac{\alpha T^2}{\beta + T} \\ E_{gNOM}(T) &= E_{g0} - \frac{\alpha \mathbf{TNOM}^2}{\mathbf{TNOM} + \beta} \\ arg1(T) &= -\frac{E_g(T)}{2kT} + \frac{E_{g300}}{2kT_0} \\ arg2(T) &= -\frac{E_{gNOM}(T)}{2k\mathbf{TNOM}} + \frac{E_{g300}}{2kT_0} \\ pbfact1(T) &= -2.0 \cdot V_t(T) \left(1.5 \cdot \ln\left(\frac{T}{T_0}\right) + q \cdot arg1(T)\right) \\ pbfact2(T) &= -2.0 \cdot V_{tnom}(T) \left(1.5 \cdot \ln\left(\frac{\mathbf{TNOM}}{T_0}\right) + q \cdot arg2(T)\right) \\ pbo(T) &= (\mathbf{VJ} - pbfact2(T)) \frac{T_0}{\mathbf{TNOM}} \\ V_J(T) &= pbfact1(T) + \frac{T}{T_0} pbo(T) \end{aligned}$$

$$
\begin{aligned}
gma_{old}(T) &= \frac{\mathbf{VJ} - pbo(T)}{pbo(T)} \\
gma_{new}(T) &= \frac{V_J(T) - pbo(T)}{pbo(T)} \\
CJ(T) &= \mathbf{CJ0}\frac{1.0 + \mathbf{M}\left(4.0 \times 10^{-4}\left(T - T_0\right) - gma_{new}(T)\right)}{1.0 + \mathbf{M}\left(4.0 \times 10^{-4}\left(\mathbf{TNOM} - T_0\right) - gma_{old}(T)\right)} \\
I_S(T) &= \mathbf{IS} \cdot \exp\left(\left(\frac{T}{\mathbf{TNOM}} - 1.0\right) \cdot \frac{\mathbf{EG}}{\mathbf{N}V_t(T)} + \frac{\mathbf{XTI}}{\mathbf{N}} \cdot \ln\left(\frac{T}{\mathbf{TNOM}}\right)\right)
\end{aligned}
$$

where, for silicon, $\alpha = 7.02 \times 10^{-4}\ eV/K$, $\beta = 1108\ ^{\circ}K$ and $E_{g0} = 1.16\ eV$.

Level=3

The level 3 diode implementation is based on [8] with radiation effects added[9].

DC Current (Level=3)

Here the I-V nature is modeled using a diffusion current I_D, a generation/recombination current I_{GR} and a breakdown current I_B

$$I = K_{HI} \cdot I_D + I_{GR} - I_B$$

where

$$I_D = \mathbf{IS} \cdot \left[\exp\left(\frac{V_{di}}{\mathbf{N}V_{th}}\right) - 1\right]$$

and the injection current factor is given by

$$K_{HI} = \begin{cases} \sqrt{\mathbf{IKF}/(\mathbf{IKF} + I_D)}, & \text{for } \mathbf{IKF} > 0 \\ 1, & \text{otherwise} \end{cases}$$

The generation/recombination current is given by

$$I_{GR} = \mathbf{ISR}\left[\left(1 - \frac{V_{di}}{\mathbf{VJ}}\right)^2 + 0.001\right]^{\mathbf{M}/2}\left[\exp\left(\frac{V_{di}}{\mathbf{NR}V_{th}}\right) - 1\right]$$

where the grading parameter $\mathbf{M}$ is $1/2$ for an abrupt junction and $1/3$ for a linearly graded junction. Lastly, the breakdown current is calculated using

$$I_B = \mathbf{IBV}\exp\left(-\frac{V_{di} + \mathbf{BV}}{\mathbf{N}V_{th}}\right)$$

Capacitance (Level=3)

The p-n diode capacitance consists of a depletion layer capacitance C_d and a diffusion capacitance C_{dif}. The first is given by

$$C_d = \begin{cases} \mathbf{CJO}\left(1 - \frac{V_{di}}{\mathbf{VJ}}\right)^{-\mathbf{M}}, & V_{di} \le \mathbf{FC} \cdot \mathbf{VJ} \\ \mathbf{CJO}(1 - \mathbf{FC})^{-(1+\mathbf{M})}\left[1 - \mathbf{FC}(1 + \mathbf{M}) + \mathbf{M}\frac{V_{di}}{\mathbf{VJ}}\right], & V_{di} > \mathbf{FC} \cdot \mathbf{VJ} \end{cases}$$

The diffusion capacitance (sometimes referred to as the transit time capacitance) is

$$C_dif = \mathbf{TT}G_d = \mathbf{TT}\frac{dI}{dV_{di}}$$

where G_d is the junction conductance and I is the diode dc current given above.

Temperature Effects (Level=3)

The diode model contains explicit temperature dependencies in the ideal diode current, the generation/recombination current and the breakdown current. Further temperature dependencies are present in the diode model via the saturation current I_S and the built-in or contact voltage V_{bi}

$$\begin{aligned} I_S(T) &= \mathbf{IS}\cdot\left(\frac{T}{T_0}\right)^{\mathbf{XTI/N}}\exp\left[\frac{\mathbf{EG}\cdot(T-T_0)}{\mathbf{N}\cdot qV_{th}T_0}\right] \\ V_{bi}(T) &= \frac{T}{T_0}\mathbf{VJ} - 2V_{th}\ln\left(\frac{T}{T_0}\right)^{1.5} - \frac{1}{q}\left[\frac{T}{T_0}E_{g0} - E_g(T)\right] \\ E_g(T) &= E_{g0} - \frac{\alpha T^2}{\beta + T} \end{aligned}$$

where, for silicon, $\alpha = 7.02\times10^{-4}\ eV/K$, $\beta = 1108\ ^\circ K$ and $E_{g0} = 1.16\ eV$.

Radiation Effects

The equations used to incorporate radiation effects into the `LEVEL=3` diode are beyond the scope of this manual. Consult [9] for a detailed analysis.

Noise

There is no noise model in this version of **Xyce**.

For a more thorough description of p-n junction physics, see [9]. For a thorough description of the U.C. Berkeley SPICE models see Reference [11].

Independent Voltage Source

General Form	`V<name> <(+) node> <(-) node> + [ [DC] <DC|TRAN value> ]` `+ [transient specification]`
Examples	`VSLOW 1 22 SIN(0.5V 1.0ma 1KHz 1ms)` `VPULSE 1 3 PULSE(-1V 1V 2ns 2ns 2ns 50ns 100ns)`
Description	Positive current flows from the positive node through the source to the negative node. DC/TRAN is the source value during a DC or Transient analysis, respectively. For a transient analysis, all independent sources can be given a time-dependent value and its value at is used for the DC analysis.
Parameters and Options	`[transient specification]` There are five predefined time-varying functions for sources: ■ `PULSE(<parameters>)` - a pulse waveform ■ `SIN(<parameters>)` - a sinusoidal waveform ■ `EXP(<parameters>)` - an exponential waveform ■ `PWL(<parameters>)` - a piecewise linear waveform ■ `SFFM(<parameters>)` - a frequency-modulated waveform

Transient Specifications

This section outlines the available transient specifications. Δt and T_F are the time step size and simulation end-time, respectively.

Pulse

```
PULSE(V1 V2 TD TR TF PW PER)
```

Parameter	Default	Unit
V1 (Initial Value)	N/A	volt
V2 (Pulse Value)	N/A	volt
TD (Delay Time)	0.0	s
TR (Rise Time)	Δt	s
TF (Fall Time)	Δt	s
PW (Pulse Width)	T_F	s
PER (Period)	T_F	s

Sine

```
SIN(V0 VA FREQ TD THETA)
```

Parameter	Default	Unit
V0 (Offset)	N/A	volt
VA (Amplitude)	N/A	volt
FREQ (Frequency)	0.0	s
TD (Delay)	Δt	s
THETA (Attenuation Factor)	Δt	s

The waveform is shaped according to the following equations:

$$V = \begin{cases} V_0, & 0 < t < T_D \\ V_0 + V_A \sin[2\pi \cdot \mathbf{FREQ} \cdot (t - T_D)] \exp[-(t - T_D) \cdot \mathbf{THETA}], & T_D < t < T_F \end{cases}$$

Exponent

```
EXP(V1 V2 TD1 TAU1 TD2 TAU2)
```

Parameter	Default	Unit
V1 (Initial Phase)	N/A	volt
VA (Amplitude)	N/A	volt
TD1 (Rise Delay Time)	0.0	s
TAU1 (Rise Time Constant)	Δt	s
TD2 (Delay Fall Time)	TD1 $+\Delta t$	s
TAU2 (Fall Time Constant)	Δt	s

The waveform is shaped according to the following equations:

$$V = \begin{cases} V_1, & 0 < t < \text{TD1} \\ V_1 + (V_2 - V_1)\{1 - \exp[-(t - \text{TD1})/\text{TAU1}]\}, & \text{TD1} < t < \text{TD2} \\ V_1 + (V_2 - V_1)\{1 - \exp[-(t - \text{TD1})/\text{TAU1}]\} & \\ \quad + (V_1 - V_2)\{1 - \exp[-(t - \text{TD2})/\text{TAU2}]\}, & \text{TD2} < t < T_2 \end{cases}$$

Piecewise Linear

```
PWL [TIME_SCALE_FACTOR=<value>]
```

```
+ [VALUE_SCALE_FACTOR=<value>] <corner points>
```

Parameter	Default	Unit
<tn> (Time at Corner)	s	none
<vn> (Voltage at Corner)	volt	none
<n> (Number of Repetitions)	positive integer, 0, or -1	none

Frequency Modulated

```
SFFM (<voff> <vampl> <fc> <mod> <fm>)
```

Parameter	Default	Unit
<voff> (Offset Voltage)	volt	none
<vampl> (Peak Voltage Amplitude)	volt	none
<fc> (Carrier Frequency)	hertz	1/**TSTOP**
<mod> (Modulation Index)		0
<fm> (Modulation Frequency)	hertz	1/**TSTOP**

The waveform is shaped according to the following equations:

$$V = \mathbf{voff} + \mathbf{vampl} \cdot \sin(2\pi \cdot \mathbf{fc} \cdot \mathbf{TIME} + \mathrm{mod} \cdot \sin(2\pi \cdot \mathbf{fm} \cdot \mathbf{TIME}))$$

where **TIME** is the current simulation time.

Independent Current Source

General Form	`I<name> <(+) node> <(-) node> + [ [DC] <DC/TRAN value> ]` `+ [transient specification]`
Examples	`ISLOW 1 22 SIN(0.5ma 1.0ma 1KHz 1ms)` `IPULSE 1 3 PULSE(-1mA 1mA 2ns 2ns 2ns 50ns 100ns)`
Description	Positive current flows from the positive node through the source to the negative node. DC/TRAN is the source value during a DC or transient analysis, respectively. For a transient analysis, all independent sources can be given a time-dependent value and its value at is used for the DC analysis.
Parameters and Option	`[transient specification]` There are five predefined time-varying functions for sources: ■ PULSE(`<parameters>`) - a pulse waveform ■ SIN(`<parameters>`) - a sinusoidal waveform ■ EXP(`<parameters>`) - an exponential waveform ■ PWL(`<parameters>`) - a piecewise linear waveform ■ SFFM(`<parameters>`) - a frequency-modulated waveform

Transient Specifications

This section outlines the available transient specifications. Δt and T_F are the time step size and simulation end-time, respectively.

Pulse

```
PULSE(I1 I2 TD TR TF PW PER)
```

Parameter	Default	Unit
I1 (Initial Value)	N/A	amp
I2 (Pulsed Value)	N/A	amp
TD (Delay Time)	0.0	s
TR (Rise Time)	Δt	s
TF (Fall Time)	Δt	s
PW (Pulse Width)	T_F	s
PER (Period)	T_F	s

Sine

```
SIN(IO IA FREQ TD THETA)
```

Parameter	Default	Unit
IO (Offset)	N/A	amp
IA (Amplitude)	N/A	amp
FREQ (Frequency)	0.0	s
TD (Delay)	Δt	s
THETA (Attenuation Factor)	Δt	s

The waveform is shaped according to the following equations:

$$I = \begin{cases} I_0, & 0 < t < T_D \\ I_0 + I_A \sin[2\pi \cdot \mathbf{FREQ} \cdot (t + T_D)] \ \exp[-(t + T_D) \cdot \mathbf{THETA}], & T_D < t < T_F \end{cases}$$

Exponent

```
EXP(I1 I2 TD1 TAU1 TD2 TAU2)
```

Parameter	Default	Unit
I1 (Initial Phase)	N/A	amp
IA (Amplitude)	N/A	amp
TD1 (Rise Delay Time)	0.0	s
TAU1 (Rise Time Constant)	Δt	s
TD2 (Delay Fall Time)	$\mathbf{TD1} + \Delta t$	s
TAU2 (Fall Time Constant)	Δt	s

The waveform is shaped according to the following equations:

$$I = \begin{cases} I_1, & 0 < t < \text{TD1} \\ I_1 + (I_2 - I_1)(1 - \exp[-(t - \text{TD1}/\text{TAU1}]), & \text{TD1} < t < \text{TD2} \\ I_1 + (I_2 - I_1)(1 - \exp[-(t - \text{TD1}/\text{TAU1}]) & \\ \quad +(I_1 - I_2)(1 - \exp[-(t - \text{TD2}/\text{TAU2}]), & \text{TD2} < t < T_2 \end{cases}$$

Piecewise Linear

```
PWL[TIME_SCALE_FACTOR=<value>] +
[VALUE_SCALE_FACTOR=<value>] (corner_points)
```

Parameter	Default	Unit
<tn> (Time at Corner)	s	none
<vn> (Voltage at Corner)	volt	none
<n> (Number of Repetitions)	positive integer, 0, or -1	none

Frequency Modulated

```
SFFM (<ioff> <iampl> <fc> <mod>  <fm>)
```

Parameter	Default	Unit
<ioff> (Offset Current)	amp	none
<iampl> (Peak Current Amplitude)	amp	none
<fc> (Carrier Frequency)	hertz	1/**TSTOP**
<mod> (Modulation Index)		0
<fm> (Modulation Frequency)	hertz	1/**TSTOP**

The waveform is shaped according to the following equations:

$$I = \mathbf{ioff} + \mathbf{iampl} \cdot \sin(2\pi \cdot \mathbf{fc} \cdot \mathbf{TIME} + \mathrm{mod} \cdot \sin(2\pi \cdot \mathbf{fm} \cdot \mathbf{TIME}))$$

Voltage Controlled Voltage Source

General Form	`E<name> <(+) node> <(-) node> <(+) controlling node> <(-) controlling node> <gain>`
Examples	`EBUFFER 1 2 10 11 5.0`
Description	A specified voltage drop elsewhere in the circuit controls the voltage-source output.
Parameters and Options	`(+)` and `(-)` nodes Output nodes. Positive current flows from the `(+)` node through the source to the `(-)` node. `<(+) controlling node>` and `<(-) controlling node>` Node pairs that define a set of controlling voltages. A given node may appear multiple times and the output and controlling nodes may be the same.

Voltage Controlled Current Source

General Form	`G<name> <(+) node> <(-) node> <(+) controlling node> <(-) controlling node> <transconductance>`
Examples	`GBUFFER 1 2 10 11 5.0`
Description	A specified voltage drop elsewhere in the circuit controls the current-source output.
Parameters and Options	`(+)` and `(-)` nodes Output nodes. Positive current flows from the `(+)` node through the source to the `(-)` node. `<(+) controlling node>` and `<(-) controlling node>` Node pairs that define a set of controlling voltages. A given node may appear multiple times and the output and controlling nodes may be the same.

Nonlinear Dependent Source

General Form	`B<name> <(+) node> <(-) node> V={ABM expression}` `B<name> <(+) node> <(-) node> I={ABM expression}`

Examples	`B1 2 0 V={sqrt(V(1))}`
	`B2 4 0 V={V(1)*TIME}`
	`B3 4 2 I={I(V1) + V(4,2)/100}`
	`B4 5 0 V={Table {V(5)}=(0,0) (1.0,2.0) (2.0,3.0) (3.0,10.0)}`

Description	The nonlinear dependent source device, also known as the B-source device, is used in analog behavioral modeling (ABM). The `(+)` and `(-)` nodes are the output nodes. Positive current flows from the `(+)` node through the source to the `(-)` node.
Comments	See Chapter 5.3, "Analog Behavioral Modeling", for more information on the Bsource device and ABM expressions, and Section 4.3, "Parameters and Expressions", for more information on expressions in general. **Note: the braces surrounding all expressions are required.**

Bipolar Junction Transistor (BJT)

General Form	`Q<name> < collector node> <base node> <emitter node>` `+ [substrate node] <model name> [area value]`

Examples	`Q2 10 2 9 PNP1`
	`Q12 14 2 0 1 NPN2 2.0`
	`Q6 VC 4 11 [SUB] LAXPNP`

Model Form	`.MODEL <model name> NPN [model parameters]` `.MODEL <model name> PNP [model parameters]`

Parameters and Options	`[substrate node]`
	Optional and defaults to ground. Since **Xyce** permits alphanumeric node names and because there is no easy way to make a distinction between these and the model names, the name (not a number) used for the substrate node must be enclosed in square brackets `[ ]`. Otherwise, nodes would be interpreted as model names. See the third example above.
	`[area value]`
	The relative device area with a default value of 1.

Comments	The BJT is modeled as an intrinsic transistor using ohmic resistances in series with the collector (RC/area), with the base (value varies with current, see BJT equations) and with the emitter (RE/area).For model parameters with optional names, such as VAF and VA (the optional name is in parentheses), either may be used.For model types NPN and PNP, the isolation junction capacitance is connected between the intrinsic-collector and substrate nodes. This is the same as in SPICE and works well for vertical IC transistor structures. For lateral IC transistor structures there is a third model, LPNP, where the isolation junction capacitance is linked between the intrinsic-base and substrate nodes.

BJT Operating Temperature

Model parameters may be assigned unique measurement temperatures using the TNOM model parameter. See BJT model parameters for more information.

Model Parameters

Table A.25 gives the available model parameters for the BJT.

Model parameters	Description	Units	Default
BF	Ideal Maximum Forward Beta		100.0
BR	Ideal Maximum Reverse Beta		1.0
CJC	Base-collector Zero-bias p-n Capacitance	farad	0.0
CJE	Base-emitter Zero-bias p-n Capacitance	farad	0.0
CJS (CCS)	Substrate Zero-bias p-n Capacitance	farad	0.0
EG	Bandgap Voltage (Barrier Height)	eV	1.11
FC	Forward-bias Depletion Capacitor Coefficient		0.5
IKF (IK)	Corner for Forward-beta High-current Roll-off	amp	1E99
IKR	Corner for Reverse-beta High-current Roll-off	amp	1E99
IRB	Current at which Rb Falls off by half	amp	0.0
IS	Transport Saturation Current	amp	1E-16
ISC	Base-collector Leakage Saturation Current	amp	0.0
ISE	Base-emitter Leakage Saturation Current	amp	0.0
ITF	Transit Time Dependency on Ic	amp	0.0
KF	Flicker Noise Coefficient		0.0
MJC	Base-collector p-n Grading Factor		0.33
MJE	Base-emitter p-n Grading Factor		0.33
MJS	Substrate p-n Grading Factor		0.0
NC	Base-collector Leakage Emission Coefficient		2.0
NE	Base-emitter Leakage Emission Coefficient		1.5
NF	Forward Current Emission Coefficient		1.0
NR	Reverse Current Emission Coefficient		1.0
PTF	Excess Phase @ $1/(2\pi \cdot \mathrm{TF})$Hz	degree	0.0
RB	Zero-bias (Maximum) Base Resistance	ohm	0.0
RBM	Minimum Base Resistance	ohm	0.0
RC	Collector Ohmic Resistance	ohm	0.0
RE	Emitter Ohmic Resistance	ohm	0.0
TF	Ideal Forward Transit Time	sec	0.0
TR	Ideal Reverse Transit Time	sec	0.0
TNOM	Parameter Measurement Temperature	°C	27.0
VAF	Forward Early Voltage	volt	1E99

Model parameters	Description	Units	Default
VAR	Reverse Early Voltage	volt	1E99
VJC (PC)	Base-collector Built-in Potential	volt	0.75
VJE (PE)	Base-emitter Built-in Potential	volt	0.75
VJS (PS)	Substrate Built-in Potential	volt	0.75
VTF	Transit Time Dependency on $\mathrm{V_{bc}}$	volt	1E99
XCJC	Fraction of **CJC** Connected Internally to **RB**		1.0
XTB	Forward and Reverse Beta Temperature Coefficient		0.0
XTF	Transit Time Bias Dependence Coefficient		0.0

Table A.25: BJT Model Parameters.

BJT Equations

The BJT implementation within **Xyce** is based on [1]. The equations in this section describe an NPN transistor. For the PNP and LPNP devices, reverse the signs of all voltages and currents. The equations use the following variables:

$$
\begin{aligned}
V_{be} &= \text{intrinsic base-intrinsic emitter voltage} \\
V_{bc} &= \text{intrinsic base-intrinsic collector voltage} \\
V_{bs} &= \text{intrinsic base-substrate voltage} \\
V_{bw} &= \text{intrinsic base-extrinsic collector voltage (quasi-saturation only)} \\
V_{bx} &= \text{extrinsic base-intrinsic collector voltage} \\
V_{ce} &= \text{intrinsic collector-intrinsic emitter voltage} \\
V_{js} &= \text{(NPN) intrinsic collector-substrate voltage} \\
&= \text{(PNP) intrinsic substrate-collector voltage} \\
&= \text{(LPNP) intrinsic base-substrate voltage} \\
V_t &= kT/q \text{ (thermal voltage)} \\
k &= \text{Boltzmann's constant} \\
q &= \text{electron charge} \\
T &= \text{analysis temperature (K)} \\
T_0 &= \text{nominal temperature (set using } \mathbf{TNOM} \text{ option)}
\end{aligned}
$$

Other variables are listed above in BJT Model Parameters.

DC Current

The BJT model is based on the Gummel and Poon model [3] where the different terminal currents are written

$$I_b = -I_{be} - I_{bc} + I_{re} + I_{rc} + S\frac{dQ_b}{dt}$$

$$
\begin{aligned}
I_e &= -I_{cc} - I_{be} + I_{re} + S\frac{dQ_{dife}}{dt} + C_{de}\frac{dV_{be}}{dt} \\
I_c &= -I_{cc} - I_{bc} + I_{rc} + S\frac{dQ_{difc}}{dt} + C_{dc}\frac{dV_{bc}}{dt}
\end{aligned}
$$

Here, Q_{dife} and Q_{difc} are the hole charges per unit area in the base affiliated with the electrons introduced across the emitter-base and collector-base junctions, respectively. Also, Q_{be} and Q_{bc} are donations to the hole charge of the base affiliated with the differences in the depletion regions of the emitter-base and collector-base junctions, respectively. Lastly, S is the cross-sectional area of the device. The intermediate currents used are defined as

$$
\begin{aligned}
-I_{be} &= \frac{\mathbf{IS}}{\mathbf{BF}}\left[\exp\left(\frac{V_{be}}{\mathbf{NF}V_{th}}\right) - 1\right] \\
-I_{cc} &= \frac{Q_{bo}}{Q_b}\mathbf{IS}\left[\exp\left(\frac{V_{be}}{\mathbf{NF}V_{th}}\right) - \exp\left(\frac{V_{bc}}{\mathbf{NF}V_{th}}\right)\right] \\
-I_{bc} &= \frac{\mathbf{IS}}{\mathbf{BR}}\left[\exp\left(\frac{V_{bc}}{\mathbf{NE}V_{th}}\right) - 1\right] \\
I_{re} &= \mathbf{ISE}\left[\exp\left(\frac{V_{be}}{\mathbf{NE}V_{th}}\right) - 1\right] \\
I_{rc} &= \mathbf{ISC}\left[\exp\left(\frac{V_{bc}}{\mathbf{NC}V_{th}}\right) - 1\right]
\end{aligned}
$$

where the last two terms are the generation/recombination currents related to the emitter and collector junctions, respectively. The charge Q_b is the majority carrier charge in the base at large injection levels and is a key difference in the Gummel-Poon model over the earlier Ebers-Moll model.

Capacitance Terms

The p-n diode capacitance consists of a depletion layer capacitance C_d and a diffusion capacitance C_{dif}. The first is given by

$$
C_d = \begin{cases} \mathbf{CJO}\left(1 - \frac{V_{di}}{\mathbf{VJ}}\right)^{-\mathbf{M}} & V_{di} \leq \mathbf{FC}\cdot\mathbf{VJ} \\ \mathbf{CJO}\,(1-\mathbf{FC})^{-(1+\mathbf{M})}\left[1 - \mathbf{FC}(1+\mathbf{M}) + \mathbf{M}\frac{V_{di}}{\mathbf{VJ}}\right] & V_{di} > \mathbf{FC}\cdot\mathbf{VJ} \end{cases}
$$

The diffusion capacitance (sometimes referred to as the transit time capacitance) is

$$
C_{dif} = \mathbf{TT}G_d = \mathbf{TT}\frac{dI}{dV_{di}}
$$

where G_d is the junction conductance and I is the diode dc current given above.

Temperature Effects

The diode model contains explicit temperature dependencies in the ideal diode current, the generation/recombination current and the breakdown current. Further temperature

dependencies are present in the diode model via the saturation current and the built-in or contact voltage

$$
\begin{aligned}
I_S(T) &= \mathbf{IS} \cdot \left(\frac{T}{T_0}\right)^{\mathbf{XTI/N}} \exp\left[\frac{\mathbf{EG} \cdot (T - T_0)}{\mathbf{N} \cdot q V_{th} T_0}\right] \\
V_{bi}(T) &= \frac{T}{T_0}\mathbf{VJ} - 2V_{th}\ln\left(\frac{T}{T_0}\right)^{1.5} - \frac{1}{q}\left[\frac{T}{T_0}E_{g0} - E_g(T)\right] \\
E_g(T) &= E_{g0} - \frac{\alpha T^2}{\beta + T}
\end{aligned}
$$

where, for silicon, $\alpha = 7.02 \times 10^{-4}\ eV/K$, $\beta = 1108\ K$ and $E_{g0} = 1.16\ eV$.

For further information on BJT models, see [3]. For a thorough description of the U.C. Berkeley SPICE models see Reference [10].

MOS Field Effect Transistor (MOSFET)

General Form

```
M<name> <drain node> <gate node> <source node>
+ <bulk/substrate node> <model name>
+ [L=<value>] [W=<value>]
+ [AD=<value>] [AS=<value>]
+ [PD=<value>] [PS=<value>]
+ [NRD=<value>] [NRS=<value>]
```

Examples

```
M5 4 12 3 0 PNOM L=20u W=10u
M3 5 13 10 0 PSTRONG
M6 7 13 10 0 PSTRONG M=2
M8 10 12 100 100 NWEAK L=30u W=20u
+ AD=288p AS=288p PD=60u PS=60u NRD=14 NRS=24
```

Model Form

```
.MODEL <model name> NMOS [model parameters]
.MODEL <model name> PMOS [model parameters]
```

Parameters and Options

`L` and `W`

The MOSFET channel length and width that are decreased to get the actual channel length and width. They may be given in the device `.MODEL` or `.OPTIONS` statements. The value in the device statement overrides the value in the model statement, which overrides the value in the `.OPTIONS` statement. Defaults for `L` and `W` may be set in the `.OPTIONS` statement. If `L` or `W` values are not given, their default value is 100 u.

`AD` and `AS`

The drain and source diffusion areas. Defaults for `AD` and `AS` can be set in the `.OPTIONS` statement. If `AD` or `AS` defaults are not set, their default value is 0.

`PD` and `PS`

The drain and source diffusion perimeters. Their default value is 0.

Parameters and Options (cont.)	NRD, NRS Multipliers (in units of squares) that can be multiplied by RSH to yield the parasitic (ohmic) resistances of the drain (RD) and source (RS), respectively. NRD, NRS default to 0. Consider a square sheet of resistive material. Analysis shows that the resistance between two parallel edges of such a sheet depends upon its composition and thickness, but is independent of its size as long as it is square. In other words, the resistance will be the same whether the square's edge is 2 mm, 2 cm, or 2 m. For this reason, the sheet resistance of such a layer, abbreviated RSH, has units of ohms per square.
Comments	The simulator provides three MOSFET device models, which differ in the formulation of the I-V characteristic. The LEVEL parameter selects among different models as shown below.

MOSFET Operating Temperature

Model parameters may be assigned unique measurement temperatures using the **TNOM** model parameter. See the MOSFET model parameters for more information.

Model Parameters

Table A.27 gives the available model parameters for the MOSFET.

All MOSFET models

The parameters shared by all MOSFET model levels are principally parasitic element values (e.g., series resistance, overlap capacitance, etc.).

Model levels 1, and 3

The DC behaviors of the level 1 and 3 MOSFET models are defined by the parameters **VTO**, **KP**, **LAMBDA**, **PHI**, and **GAMMA**. The simulator calculates these if the process parameters (e.g., **TOX**, and **NSUB**) are specified, but these are always overridden by any user-defined values. The **VTO** value is positive (negative) for modeling the enhancement mode and negative (positive) for the depletion mode of N-channel (P-channel) devices.

For MOSFETs, the capacitance model enforces charge conservation, influencing just the Level 1 and 3 models.

Effective device parameter lengths and widths are calculated as follows:

$$P_i = P_0 + P_L/L_e + P_W/W_e$$

where

$$\begin{array}{rcl} L_e & = & \text{effective length} \quad = \quad \mathbf{L} - (2 \cdot \mathbf{LD}) \\ W_e & = & \text{effective width} \quad = \quad \mathbf{W} - (2 \cdot \mathbf{WD}) \end{array}$$

See `.MODEL` (model definition) for more information.

Model level 9 (BSIM3 version 3.1)

The University of California, Berkeley BSIM3 model is a physical-based model with a large number of dependencies on essential dimensional and processing parameters. It incorporates the key effects that are critical in modeling deep-submicrometer MOSFETs. These include threshold voltage reduction, nonuniform doping, mobility reduction due to the vertical field, bulk charge effect, carrier velocity saturation, drain-induced barrier lowering (DIBL), channel length modulation (CLM), hot-carrier-induced output resistance reduction, subthreshold conduction, source/drain parasitic resistance, substrate current induced body effect (SCBE) and drain voltage reduction in LDD structure.

The BSIM3 Version 3.1 model is a deep submicron MOSFET model with several major enhancements (over earlier versions). These include a single I-V formula used to define the current and output conductance for operating regions, improved narrow width device modeling, a superior capacitance model with improved short and narrow geometry models, a new relaxation-time model to better transient modeling and enhanced model fitting of assorted W/L ratios using a single parameter set. This version preserves the large number of integrated dependencies on dimensional and processing parameters of the Version 2 model. For further information, see Reference [11].

Additional notes

1. If any of the following BSIM3 3.1 model parameters are not specified, they are computed via the following:

   If $\mathbf{VTHO}$ is not specified, then:

   $$\mathbf{VTHO} = \mathbf{VFB} + \phi_s \mathbf{K1} \sqrt{\phi_s}$$

   where:

   $$\mathbf{VFB} = -1.0$$

   If $\mathbf{VTHO}$ is given, then:

   $$\begin{array}{rcl} \mathbf{VFB} & = & \mathbf{VTHO} - \phi_s + \mathbf{K1} \sqrt{phi_s} \\ \mathbf{VBX} & = & \phi_s - \dfrac{q \cdot \mathbf{NCH} \cdot \mathbf{XT}^2}{2\varepsilon_{si}} \\ \mathbf{CF} & = & \left(\dfrac{2\varepsilon_{ox}}{\pi} \right) \ln \left(1 + \dfrac{1}{4 \times 10^7 \cdot \mathbf{TOX}} \right) \end{array}$$

   where:

   $E_g(T) =$ the energy bandgap at temperature $T = 1.16 - \dfrac{T^2}{7.02 \times 10^4 (T + 1108)}$

2. If $\mathbf{K1}$ and $\mathbf{K2}$ are not given then they are computed via the following:

$$\begin{aligned}\mathbf{K1} &= \mathbf{GAMMA2} - 2\cdot\mathbf{K2}\sqrt{\phi_s - \mathbf{VBM}} \\ \mathbf{K2} &= \frac{(\mathbf{GAMMA1}-\mathbf{GAMMA2})(\sqrt{\phi_s-\mathbf{VBX}}-\sqrt{\phi_s})}{2\sqrt{\phi_s}(\sqrt{\phi_s-\mathbf{VBM}}-\sqrt{\phi_s})+\mathbf{VBM}}\end{aligned}$$

where:

$$\begin{aligned}\phi_s &= 2V_t \ln\left(\frac{\mathbf{NCH}}{n_i}\right) \\ V_t &= kT/q \\ n_i &= 1.45\times 10^{10}\left(\frac{T}{300.15}\right)^{1.5}\exp\left(21.5565981 - \frac{E_g(T)}{2V_t}\right)\end{aligned}$$

3. If $\mathbf{NCH}$ is not specified and $\mathbf{GAMMA1}$ is, then:

$$\mathbf{NCH} = \frac{\mathbf{GAMMA1^2}\times\mathbf{COX^2}}{2q\varepsilon_{si}}$$

If $\mathbf{GAMMA1}$ and $\mathbf{NCH}$ *are not* specified, then $\mathbf{NCH}$ defaults to $1.7\times10^{23}\ m^{-3}$ and $\mathbf{GAMMA1}$ is computed using $\mathbf{NCH}$:

$$\mathbf{GAMMA1} = \frac{\sqrt{2q\varepsilon_{si}\cdot\mathbf{NCH}}}{\mathbf{COX}}$$

If $\mathbf{GAMMA2}$ is not specified, then:

$$\mathbf{GAMMA2} = \frac{\sqrt{2q\varepsilon_{si}\cdot\mathbf{NSUB}}}{\mathbf{COX}}$$

4. If $\mathbf{CGSO}$ is not specified and $\mathbf{DLC} > 0$, then:

$$\mathbf{CGSO} = \begin{cases} 0, & ((\mathbf{DLC}\cdot\mathbf{COX}) - \mathbf{CGSL}) < 0 \\ 0.6\cdot\mathbf{XJ}\cdot\mathbf{COX}, & ((\mathbf{DLC}\cdot\mathbf{COX}) - \mathbf{CGSL}) \geq 0 \end{cases}$$

5. If $\mathbf{CGDO}$ is not specified and $\mathbf{DLC} > 0$, then:

$$\mathbf{CGDO} = \begin{cases} 0, & ((\mathbf{DLC}\cdot\mathbf{COX}) - \mathbf{CGSL}) < 0 \\ 0.6\cdot\mathbf{XJ}\cdot\mathbf{COX}, & ((\mathbf{DLC}\cdot\mathbf{COX}) - \mathbf{CGSL}) \geq 0 \end{cases}$$

Model parameters	Description	Units	Default
	All Levels		
AF	Flicker Noise Exponent		1.0
CBD	Zero-bias Bulk-drain p-n Capacitance	farad (F)	0.0
CBS	Zero-bias Bulk-source p-n Capacitance	F	0.0
CGBO	Gate-bulk Overlap Capacitance/Channel Length	F	0.0

Model parameters	Description	Units	Default
CGDO	Gate-drain Overlap Capacitance/Channel Width	F	0.0
CGSO	Gate-source Overlap Capacitance/Channel Width	F	0.0
CJ	Bulk p-n Zero-bias Bottom Capacitance/Area	F/m^2	0.0
CJSW	Bulk p-n Zero-bias Sidewall Capacitance/Area	F/m^2	0.0
FC	Bulk p-n Forward-bias Capacitance Coefficient		0.5
IS	Bulk p-n Saturation Current	amp	1E-14
JS	Bulk p-n Saturation Current/Area	amp/m^2	0.0
KF	Flicker Noise Coefficient		0.0
L	Channel Length	m	**DEFL**
LEVEL	Model Index		**1**
MJ	Bulk p-n Bottom Grading Coefficient		0.5
MJSW	Bulk p-n Sidewall Grading Coefficient		0.33
PB	Bulk p-n Bottom Potential	volt	0.8
RD	Drain Ohmic Resistance	ohm	0.0
RS	Source Ohmic Resistance	ohm	0.0
RSH	Drain, Source Diffusion Sheet Resistance	ohm	0.0
TT	Bulk p-n Transit Time	second (s)	0.0
TNOM	Parameter Measurement Temperature	°C	27.0
W	Channel Width	m	**DEFW**
	Levels 1and 3		
DELTA	Width Effect on Threshold		0.0
ETA	Static Feedback (Level 3)		0.0
GAMMA	Bulk Threshold Parameter	volt $(V)^{1/2}$	
KP	Transconductance Coefficient	amp/V^2	2.0E-5
KAPPA	Saturation Field Factor (Level 3)		0.2
LAMBDA	Channel-length Modulation (Levels 1)	V^{-1}	0.0
LD	Lateral Diffusion (Length)	m	0.0
NFS	Fast Surface State Density	cm^{-2}	0.0
NSS	Surface State Density	cm^{-2}	
NSUB	Substrate Doping Density	cm^{-3}	0.0
PHI	Surface Potential	V	0.6

Model parameters	Description	Units	Default
THETA	Mobility Modulation (Level 3)	V^{-1}	0.0
TOX	Oxide Thickness	m	
TPG	Gate Material Type: $+1$ = opposite of substrate -1 = same as substrate 0 = aluminum		+1
UTRA	(Not Used) Mobility Degradation Transverse Field Coefficient		0.0
UO	Surface Mobility	$cm^{-2}/(V \cdot s)$	600
VMAX	Maximum Drift Velocity	m/s	0.0
VTO	Zero-bias Threshold Voltage	V	0.0
WD	Lateral Diffusion (Width)	m	0.0
XJ	Metallurgical Junction Depth (Level 3)	m	0.0
XQC	Fraction of Channel Charge Attributed to Drain		1.0
Level 9: Control Parameters			
CAPMOD	Flag for the Short-channel Capacitance Model		2.0
MOBMOD	Mobility Model Selector		1
NOIMOD	Flag for Noise Model		1
NQSMOD	Flag for NQS Model		0
PARAMCHK	Flag for Model Parameter Checking		0
Level 9: Capacitance Parameters			
CF	Fringing Field Capacitance	F/m	
CKAPPA	Coefficient for Lightly Doped Region Overlap Capacitance Fringing Field Capacitance	F/m	0.6
CLC	Constant Term for the Short-Channel Model	m	0.1E-6
CLE	Exponential Term for the Short-Channel Model		0.6
CGBO	Gate-bulk Overlap Capacitance per Unit Channel Length	F/m	0.0
CGDL	Light-doped Drain-gate Region Overlap Capacitance	F/m	0.0
Level 9: Capacitance Parameters (cont.)			
CGDO	Non-LDD Region Drain-gate Overlap Capacitance per Unit Channel Length	F/m	

Model parameters	Description	Units	Default
CGSL	Light-doped Source-gate Region Overlap Capacitance	$\mathrm{F/m}$	0.0
CGSO	Non-LDD Region Source-gate Overlap Capacitance per Unit Channel Length	$\mathrm{F/m}$	
CJ	Bottom Junction Capacitance per Unit Area	$\mathrm{F/m^2}$	5.0E-4
CJSW	Source/Drain Side Junction Capacitance per Unit Periphery	$\mathrm{F/m}$	5.0E-10
CJSWG	Source/Drain Gate Sidewall Junction Capacitance per Unit Width	$\mathrm{F/m}$	**CJSW**
DLC	Length Offset Fitting Parameter from C-V	m	**LINT**
DWC	Width Offset Fitting Parameter from C-V	m	**WINT**
MJ	Bottom Junction Capacitance Grading Coefficient		0.5
MJSW	Source/Drain Side Junction Capacitance Grading Coefficient		0.33
MJSWG	Source/Drain Gate Sidewall Junction Capacitance Grading Coefficient		**MJSW**
PB	Bottom Built-in Potential	V	1.0
PBSW	Source/Drain Side Junction Built-in Potential	V	1.0
PBSWG	Source/Drain Gate Sidewall Junction Built-in Potential	V	**PBSW**
VFBCV	Flat-band Voltage Parameter (for **CAPMOD** $=0$ only)	V	-1.0
XPART	Charge Partitioning Rate Flag		0.0
Level 9: Bin Description Parameters			
BINUNIT	Bin Unit Scale Selector		1.0
LMAX	Maximum Channel Length	m	1.0
LMIN	Minimum Channel Length	m	0.0
WMAX	Maximum Channel Width	m	1.0
WMIN	Minimum Channel Width	m	0.0
Level 9: DC Parameters			
A0	Bulk Charge Effect Coefficient for Channel Length		1.0
A1	First Non-saturation Effect Parameter	$\mathrm{V^{-1}}$	0.0
A2	Second Non-saturation Factor		1.0
AGS	Gate-bias Coefficient of Abulk	$\mathrm{V^{-1}}$	0.0

Model parameters	Description	Units	Default
ALPHA0	First Parameter of Impact-ionization Current	$\mathrm{m/V}$	0.0
B0	Bulk Charge Effect Coefficient for Channel Width	m	0.0
B1	Bulk Charge Effect Width Offset	m	0.0
BETA0	Second Parameter of Impact-ionization Current	V	30.0
CDSC	Drain/Source to Channel Coupling Capacitance	$\mathrm{F/m^2}$	2.4E-4
CDSCB	Body-bias Sensitivity of **CDSC**	$\mathrm{F/Vm^2}$	0.0
CDSCD	Drain-bias Sensitivity of **CDSC**	$\mathrm{F/Vm^2}$	0.0
CIT	Interface Trap Capacitance	$\mathrm{F/m^2}$	0.0
DELTA	Effective **Vds** Parameter	V	0.01
DROUT	L-dependence Coefficient of the **DIBL** Correction Parameter in **Rout**		0.56
DSUB	**DIBL** Coefficient Exponent in Subthreshold Region		**DROUT**
DVT0	First Coefficient of Short-channel Effect on Threshold Voltage		2.2
DVT0W	First Coefficient of Narrow-width Effect on Threshold Voltage for Small-channel Length	$\mathrm{m^{-1}}$	0.0
DVT1	Second Coefficient of Short-channel Effect on Threshold Voltage		0.53
DVT2	Body-bias Coefficient of Short-channel Effect on Threshold Voltage	$\mathrm{V^{-1}}$	-0.032
DVTW1	Second Coefficient of Narrow-channel Effect on Threshold Voltage for Small Channel Length	$\mathrm{m^{-1}}$	5.3E6
DVTW2	Body-bias Coefficient of Narrow-width Effect for Small Channel Length	$\mathrm{V^{-1}}$	-0.032
DWB	Coefficient of Substrate Body Bias Dependence of **Weff**	$\mathrm{m/V^{1/2}}$	0.0
DWG	Coefficient of Gate Dependence of **Weff**	$\mathrm{m/V}$	0.0
ETA0	**DIBL** Coefficient in Subthreshold Region		0.08
ETAB	Body-bias Coefficient for the Subthreshold **DIBL** Effect	$\mathrm{V^{-1}}$	-0.07
JS	Source-drain Junction Saturation Current per Unit Area	$\mathrm{amp/m^2}$	1.0E-4

Model parameters	Description	Units	Default
JSW	Sidewall Saturation Current per Unit Length	$\mathrm{amp/m}$	0.0
K1	First-order Body Effect Coefficient	$\mathrm{V}^{1/2}$	0.5
K2	Second-order Body Effect Coefficient		0.0
K3	Narrow Width Coefficient		80.0
K3B	Body Effect Coefficient of **K3**	V^{-1}	0.0
KETA	Body-bias Coefficient of Bulk Charge Effect	V^{-1}	-0.047
LINT	Length Offset Fitting Parameter from I-V Without Bias	m	0.0
NFACTOR	Subthreshold Swing Factor		1.0
NGATE	Poly Gate Doping Concentration	cm^{-3}	0.0
NLX	Lateral Non-uniform Doping Parameter	m	1.74E-7
PCLM	Channel Length Modulation Parameter		1.3
PDIBLC1	First Output Resistance **DIBL** Effect Correction Parameter		0.39
PDIBLC2	Second Output Resistance **DIBL** Effect Correction Parameter		0.0086
PDIBLCB	Body Effect Coefficient of **DIBL** Correction Parameter	V^{-1}	0.0
PRWB	Body Effect Coefficient of **RDSW**	$\mathrm{V}^{-1/2}$	0.0
PRWG	Gate-bias Effect Coefficient of **RDSW**	V^{-1}	0.0
PSCBE1	First Substrate Current Body Effect Parameter	$\mathrm{V/m}$	4.24E8
PSCBE2	Second Substrate Current Body Effect Parameter	$\mathrm{V/m}$	1.0E-5
PVAG	Gate Dependence of Early Voltage		0.0
RDSW	Parasitic Resistance per Unit Width	ohm-$\mu\mathrm{m}^{\mathrm{WR}}$	0.0
RSH	Source-drain Sheet Resistance	$\mathrm{ohm/}$ square?	0.0
U0	Mobility at Temp $=$ **TNOM** **NMOS** **PMOS**	$\mathrm{cm}^2/\mathrm{V}\cdot\mathrm{s}$	670.0 250.0
UA	First-order Mobility Degradation Coefficient	$\mathrm{m/V}$	2.25E-9
UB	Second-order Mobility Degradation Coefficient	$(\mathrm{m/V})^2$	5.87E-19
UC	Body Effect of Mobility Degradation Coefficient **MOBMOD** $=1$ **MOBMOD** $=3$	$\mathrm{m/V}^2$ $1/\mathrm{V}$	-4.65E-11 -0.046
VBM	Maximum Applied Body-bias in Threshold Voltage Calculation	V	-3.0

Model parameters	Description	Units	Default
VOFF	Offset Voltage in the Subthreshold Region at Large W and L	V	-0.08
VSAT	Saturation Velocity at Temp = **TNOM**	m/s	8.0E4
VTHO	Threshold Voltage at **Vbs** = 0 for Large **L**	V	0.7 (**NMOS**) -0.7 (**PMOS**)
WO	Narrow-width Parameter	m	2.5E-6
WINT	Width-offset Fitting Parameter from I-V Without Bias	m	0.0
WR	Width-offset from Weff for **Rds** Calculation		1.0
Level 9: Flicker Noise Parameters			
AF	Frequency Exponent		1.0
EF	Flicker Exponent		1.0
EM	Saturation Field	V/m	4.1E7
KF	Flicker Noise Parameter		0.0
NOIA	Noise Parameter A **NMOS** **PMOS**		1.0E20 9.9E18
NOIB	Noise Parameter B **NMOS** **PMOS**		5.0E4 2.4E3
NOIC	Noise Parameter C **NMOS** **PMOS**		-1.4E-12 1.4E-12
Level 9: NQS Parameter			
ELM	Elmore Constant of the Channel		5.0
Level 9: Process Parameters			
GAMMA1	Body Effect Coefficient Near the Surface	$V^{1/2}$	
GAMMA2	Body Effect Coefficient in the Bulk	$V^{1/2}$	
NCH	Channel Doping Concentration	cm^{-3}	1.7E17
NSUB	Substrate Doping Concentration	cm^{-3}	6.0E16
TOX	Gate-oxide Thickness	m	1.5E-8
VBX	**Vbs** at Which the Depletion Region = **XT**	V	
XJ	Junction Depth	m	1.5E-7
XT	Doping Depth	m	1.55E-7
Level 9: Temperature Parameters			
AT	Temperature Coefficient for Saturation Velocity	m/s	3.3E4
KT1	Temperature Coefficient for Threshold Voltage	V	-0.11

Model parameters	Description	Units	Default
KT1L	Channel Length Dependence of the Temperature Coefficient for Threshold Voltage	$\mathrm{V \cdot m}$	0.0
KT2	Body-bias Coefficient of Threshold Voltage Temperature Effect		0.022
NJ	Emission Coefficient of Junction		1.0
PRT	Temperature Coefficient for **RDSW**	$\mathrm{ohm\text{-}\mu m}$	0.0
TNOM	Temperature at which Parameters are Extracted	$^{\circ}\mathrm{C}$	27.0
UA1	Temperature Coefficient for **UA**	$\mathrm{m/V}$	4.31E-9
UB1	Temperature Coefficient for **UB**	$(\mathrm{m/V})^2$	-7.61E-18
UC1	Temperature Coefficient for **UC** **MOBMOD** $=1$ or 2 **MOBMOD** $=3$	$\mathrm{m/V^2}$ $1/\mathrm{V}$	-5.6E-11 -0.056
UTE	Mobility Temperature Exponent		-1.5
XT1	Junction Current Temperature Exponent Coefficient		3.0
Level 9: W and L Parameters			
LL	Coefficient of Length Dependence for Length Offset	$\mathrm{m^{LLN}}$	0.0
LLN	Power of Length Dependence for Length Offset		1.0
LW	Coefficient of Width Dependence for Length Offset	$\mathrm{m^{LWN}}$	0.0
LWL	Coefficient of Length and Width Cross Term for Length Offset	$\mathrm{m^{LWN+LLN}}$	0.0
LWN	Power of Width Dependence for Length Offset		1.0
WL	Coefficient of Length Dependence for Width Offset	$\mathrm{m^{WLN}}$	0.0
WW	Coefficient of Width Dependence for Width Offset	$\mathrm{m^{WWN}}$	0.0
WWL	Coefficient of Length and Width Cross Term for Width Offset	$\mathrm{m^{WWN+WLN}}$	0.0
WWN	Power of Width Dependence of Width Offset		1.0

Table A.27: MOSFET Model Parameters.

MOSFET Equations

The following equations define an N-channel MOSFET. The P-channel devices use a reverse the sign for all voltages and currents. The equations use the following variables:

V_{bs} = intrinsic substrate-intrinsic source voltage
V_{bd} = intrinsic substrate-intrinsic drain voltage
V_{ds} = intrinsic drain-substrate source voltage
V_{dsat} = saturation voltage
V_{gs} = intrinsic gate-intrinsic source voltage
V_{gd} = intrinsic gate-intrinsic drain voltage
V_t = kT/q (thermal voltage)
V_{th} = threshold voltage
C_{ox} = the gate oxide capacitance per unit area
f = noise frequency
k = Boltzmann's constant
q = electron charge
$Leff$ = effective channel length
$Weff$ = effective channel width
T = analysis temperature ($^\circ K$)
T_0 = nominal temperature (set using TNOM option)

Other variables are listed in the BJT Model Parameters Table A.25.

All Levels

$$
\begin{aligned}
I_g &= \text{gate current} = 0 \\
I_b &= \text{bulk current} = I_{bs} + I_{bd}
\end{aligned}
$$

where

$$
I_{bs} = \text{bulk-source leakage current} = I_{ss}\left(e^{V_{bs}/(N-V_t)} - 1\right)
$$

$$
I_{ds} = \text{bulk-drain leakage current} = I_{ds}\left(e^{V_{bd}/(N-V_t)} - 1\right)
$$

where

if

$$\mathbf{JS} = 0, \text{ or } \mathbf{AS} = 0 \text{ or } \mathbf{AD} = 0$$

then

$$I_{ss} = \mathbf{IS}$$
$$I_{ds} = \mathbf{IS}$$

else

$$I_{ss} = \mathrm{AS} \times \mathbf{JS} + \mathrm{PS} \times \mathbf{JSSW}$$
$$I_{ds} = \mathrm{AD} \times \mathbf{JS} + \mathrm{PD} \times \mathbf{JSSW}$$
$$I_d = \text{drain current} = I_{drain} - I_{bd}$$
$$I_s = \text{source current} = -I_{drain} - I_{bs}$$

Level 1: Idrain

Normal Mode: $V_{ds} > 0$

Case 1

For cutoff region: $V_{gs} - V_{to} < 0$

$$I_{drain} = 0$$

Case 2

For linear region: $V_{ds} < V_{gs} - V_{to}$

$$I_{drain} = (W/L)(\mathbf{KP}/2)(1 + \mathbf{LAMBDA} \times V_{ds})V_{ds}(2(V_{gs} - V_{to}) - V_{ds})$$

Case 3

For saturation region: $0 \leq V_{gs} - V_{to} \leq V_{ds}$

$$I_{drain} = (W/L)(\mathbf{KP}/2)(1 + \mathbf{LAMBDA} \cdot V_{ds})(V_{gs} - V_{to})^2$$

where

$$V_{to} = \mathbf{VTO} + \mathbf{GAMMA} \cdot \left((\mathbf{PHI} - V_{bs})^{1/2}\right)^{1/2}$$

Inverted Mode: $V_{ds} < 0$

Here, simply switch the source and drain in the normal mode equations given above.

Level 3: Idrain

See Reference [12] below for detailed information.

Capacitance

Level 1 and 3

C_{bs} = bulk-source capacitance = area cap. + sidewall cap. + transit time cap.

C_{bd} = bulk-drain capacitance = area cap. + sidewall cap. + transit time cap.

where

$$
\begin{aligned}
&if \\
&\qquad \textbf{CBS} = 0 \text{ and } \textbf{CBD} = 0 \\
&then \\
&\qquad C_{bs} = \mathrm{AS} \cdot \mathbf{CJ} \cdot C_{bsj} + \mathrm{PS} \cdot \mathbf{CJSW} \cdot C_{bss} + \mathbf{TT} \cdot G_{bs} \\
&\qquad C_{bd} = \mathrm{AD} \cdot \mathbf{CJ} \cdot C_{bdj} + \mathrm{PD} \cdot \mathbf{CJSW} \cdot C_{bds} + \mathbf{TT} \cdot G_{ds} \\
&else \\
&\qquad C_{bs} = \mathrm{CBS} \cdot C_{bsj} + \mathrm{PS} \cdot \mathbf{CJSW} \cdot C_{bss} + \mathbf{TT} \cdot G_{bs} \\
&\qquad C_{bd} = \mathrm{CBD} \cdot C_{bdj} + \mathrm{PD} \cdot \mathbf{CJSW} \cdot C_{bds} + \mathbf{TT} \cdot G_{ds} \\
&where \\
&\qquad G_{bs} = \text{DC bulk-source conductance} = dI_{bs}/dV_{bs} \\
&\qquad G_{bd} = \text{DC bulk-drain conductance} = dI_{bd}/dV_{bd}
\end{aligned}
$$

$$
\begin{aligned}
&if \\
&\qquad V_{bs} \leq \mathbf{FC} \cdot \mathbf{PB} \\
&then \\
&\qquad C_{bsj} = (1 - V_{bs}/\mathbf{PB})^{-\mathbf{MJ}} \\
&\qquad C_{bss} = (1 - V_{bs}/\mathbf{PBSW})^{-\mathbf{MJSW}} \\
&if \\
&\qquad V_{bs} > \mathbf{FC} \cdot \mathbf{PB} \\
&then \\
&\qquad C_{bsj} = (1 - \mathbf{FC})^{-(1+\mathbf{MJ})}(1 - \mathbf{FC}(1 + \mathbf{MJ}) + \mathbf{MJ} \cdot V_{bs}/\mathbf{PB}) \\
&\qquad C_{bss} = (1 - \mathbf{FC})^{-(1+\mathbf{MJSW})}(1 - \mathbf{FC}(1 + \mathbf{MJSW}) + \mathbf{MJSW} \cdot V_{bs}/\mathbf{PBSW}) \\
&if \\
&\qquad V_{bd} \leq \mathbf{FC} \cdot \mathbf{PB} \\
&then \\
&\qquad C_{bdj} = (1 - V_{bd}/\mathbf{PB})^{-\mathbf{MJ}} \\
&\qquad C_{bds} = (1 - V_{bd}/\mathbf{PBSW})^{-\mathbf{MJSW}} \\
&if \\
&\qquad V_{bd} > \mathbf{FC} \cdot \mathbf{PB} \\
&then \\
&\qquad C_{bdj} = (1 - \mathbf{FC})^{-(1+\mathbf{MJ})}(1 - \mathbf{FC}(1 + \mathbf{MJ}) + \mathbf{MJ} \cdot V_{bd}/\mathbf{PB}) \\
&\qquad C_{bds} = (1 - \mathbf{FC})^{-(1+\mathbf{MJSW})}(1 - \mathbf{FC}(1 + \mathbf{MJSW}))
\end{aligned}
$$

$$
\begin{aligned}
&C_{gs} = \text{gate-source overlap capacitance} = \mathbf{CGSO} \cdot \mathbf{W} \\
&C_{gd} = \text{gate-drain overlap capacitance} = \mathbf{CGDO} \cdot \mathbf{W} \\
&C_{gb} = \text{gate-bulk overlap capacitance} = \mathbf{CGBO} \cdot \mathbf{L}
\end{aligned}
$$

Temperature Effects

All Levels

$$\mathbf{IS}(T) = \mathbf{IS} \cdot \exp\left(E_g(T_0) \cdot T/T_0 - E_g(T)\right)/V_t$$
$$\mathbf{JS}(T) = \mathbf{JS} \cdot \exp\left(E_g(T_0) \cdot T/T_0 - E_g(T)\right)/V_t$$
$$\mathbf{JSSW}(T) = \mathbf{JSSW} \cdot \exp\left(E_g(T_0) \cdot T/T_0 - E_g(T)\right)/V_t$$
$$\mathbf{PB}(T) = \mathbf{PB} \cdot T/T_0 - 3V_t \ln(T/T_0) - E_g(T_0) \cdot T/T_0 + E_g T$$
$$\mathbf{PBSW}(T) = \mathbf{PBSW} \cdot T/T_0 - 3V_t \ln(T/T_0) - E_g(T_0) \cdot T/T_0 + E_g T$$
$$\mathbf{PHI}(T) = \mathbf{PHI} \cdot T/T_0 - 3V_t \ln(T/T_0) - E_g(T_0) \cdot T/T_0 + E_g T$$

where

$$E_g(T) = \text{silicon bandgap energy} = 1.16 - 0.000702T^2/(T + 1108)$$
$$\mathbf{CBD}(T) = \mathbf{CBD} \cdot (1 + \mathbf{MJ} \cdot (0.0004(T - T_0) + (1 - \mathbf{PB}(T)/\mathbf{PB})))$$
$$\mathbf{CBS}(T) = \mathbf{CBS} \cdot (1 + \mathbf{MJ} \cdot (0.0004(T - T_0) + (1 - \mathbf{PB}(T)/\mathbf{PB})))$$
$$\mathbf{CJ}(T) = \mathbf{CJ} \cdot (1 + \mathbf{MJ} \cdot (0.0004(T - T_0) + (1 - \mathbf{PB}(T)/\mathbf{PB})))$$
$$\mathbf{CJSW}(T) = \mathbf{CJSW} \cdot (1 + \mathbf{MJSW} \cdot (0.0004(T - T_0) + (1 - \mathbf{PB}(T)/\mathbf{PB})))$$
$$\mathbf{KP}(T) = \mathbf{KP} \cdot (T/T_0)^{-3/2}$$
$$\mathbf{UO}(T) = \mathbf{UO} \cdot (T/T_0)^{-3/2}$$
$$\mathbf{MUS}(T) = \mathbf{MUS} \cdot (T/T_0)^{-3/2}$$
$$\mathbf{MUZ}(T) = \mathbf{MUZ} \cdot (T/T_0)^{-3/2}$$
$$\mathbf{X3MS}(T) = \mathbf{X3MS} \cdot (T/T_0)^{-3/2}$$

For a thorough description of MOSFET models see [13, 12, 14, 15, 16, 10, 11, 17, 18, 19].

Voltage-controlled Switch

General Form	S<name> <(+) switch node> <(-) switch node> + <(+) controlling node> <(-) controlling node> + <model name>
Examples	S1 21 23 12 10 SMOD1 SSET 15 10 1 13 SRELAY
Description	The voltage-controlled switch is a particular type of voltage-controlled resistor. This model is designed to help reduce numerical issues. See Special considerations below.
Comments	The resistance between the <(+) switch node> and the <(-) switch node> is dependent on the voltage between the <(+) controlling node> and the <(-) controlling node>. The resistance changes in a continuous manner between the RON and ROFF model parameters. A resistance of 1/GMIN (GMIN can be specified as a .OPTIONS DEVICE parameter) is attached across the controlling nodes to prevent them from floating. Even though evaluating the switch model is computationally inexpensive, for transient analysis, **Xyce** steps through the transition section using small time- steps in order to accurately calculate the waveform. Thus, a circuit with many switch transitions can result in lengthy run times.

Model Parameters

Table A.32 gives the available model parameters for the voltage-controlled switch.

Model parameters	Description	Units	Default
ROFF	Off Resistance	ohm	1.0×10^6
RON	On Resistance	ohm	1.0
VOFF	Control Voltage for Off State	volt	0.0
VON	Control Voltage for 0n State	volt	1.0

Table A.32. Voltage-controlled Switch Model Parameters.

Special Considerations

- Due to numerical limitations, **Xyce** can only manage a dynamic range of approximately 12 decades. Thus, it is recommended the user limit the ratio $\mathbf{ROFF/RON}$ to less than 10^{12}.

- Furthermore, it is a good idea to limit the narrowness of the transition region. This is because in the transition region, the switch has gain and the narrower the region, the

higher the gain and the more potential for numerical problems. The smallest value allowed for $\|\mathbf{VON} - \mathbf{VOFF}\|$ is 1×10^{-12}.

Voltage-controlled switch equations

The equations in this section use the following variables:

$$
\begin{array}{rcll}
R_s & = & \text{switch resistance} & \\
V_c & = & \text{voltage across control nodes} & \\
L_m & = & \text{log-mean of resistor values} & = \ln\left(\sqrt{\mathbf{RON} \cdot \mathbf{ROFF}}\right) \\
L_r & = & \text{log - ratio of resistor values} & = \ln\left(\mathbf{RON}/\mathbf{ROFF}\right) \\
V_d & = & \text{difference of control voltages} & = \mathbf{VON} - \mathbf{VOFF}
\end{array}
$$

Switch Resistance

To compute the switch resistance, **Xyce** first calculates the "switch state" S as $S = (V_c - \mathbf{VOFF})/V_d$. The switch resistance is then:

$$
R_s = \begin{cases} \mathbf{RON}, & S \geq 1.0 \\ \mathbf{ROFF}, & S \leq 0.0 \\ \exp\left(L_m + 0.75L_r(2S-1) - 0.25L_r(2S-1)^3\right), & 0 < S < 1 \end{cases}
$$

Lossless (Ideal) Transmission Line

General Form	`T<name> <A port (+) node> <A port (-) node>`
	`+ <B port (+) node> <B port (-) node> [model name]`
	`+ Z0=<value> [TD=<value>] [F=<value> [NL=<value>]]`

Examples	`Tline inp inn outp outn Z0=50 TD=1us` `Tline2 inp inn outp outn Z0=50 F=1meg NL=1.0`
Description	The lossless transmission line device is a two port (`A` and `B`), bi-directional delay line. The `(+)` and `(-)` nodes define the polarity of a positive voltage at a port.
Comments	`Z0` is the characteristic impedance. The transmission line's length is specified by either `TD` (a delay in seconds) or by `F` and `NL` (a frequency and relative wavelength at `F`). `NL` defaults to 0.25 (`F` is the quarter-wave frequency). If `F` is given, the time delay is computed as $\frac{NL}{F}$. While both `TD` and `F` are optional, at least one of them must be given.

Model Parameters

Table A.34 gives the available model parameters for the lossless transmission line.

Model parameters	Description	Units	Default
`Z0`	Characteristic impedance	ohm	-
`TD`	Transmission delay	second	-
`F`	Frequency for **NL**	Hz	-
`NL`	Relative wavelength	-	0.25

Table A.34. Lossless (Ideal) Transmission Line Parameters.

Subcircuit

A subcircuit can be introduced into the circuit netlist using the specified nodes to substitute for the argument nodes in the definition. It provides a building block of circuitry to be defined a single time and subsequently used multiple times in the overall circuit netlists.

General Form	`X<name> [node]* <subcircuit name> [PARAMS: [<name> = <value>]*]`

Examples	`X12 100 101 200 201 DIFFAMP`
	`XBUFF 13 15 UNITAMP`
	`XFOLLOW IN OUT VCC VEE OUT OPAMP`
	`XFELT 1 2 FILTER PARAMS: CENTER=200kHz`
	`XNANDI 25 28 7 MYPWR MYGND PARAMS: IO_LEVEL=2`

Parameters and Options	`<subcircuit name>`
	The name of the subcircuit's definition.
	`PARAMS:`
	Passed into subcircuits as arguments and into expressions inside the subcircuit.

Comments	There must be an equal number of nodes in the subcircuit call and in its definition. Subcircuit references may be nested to any level. However, the nesting cannot be circular. For example, if subcircuit `A`'s definition includes a call to subcircuit `B`, then subcircuit `B`'s definition cannot include a call to subcircuit `A`.

PDE Devices

Semiconductor device simulation, which is based on a coupled set of partial differential equations (PDE's) is supported in **Xyce**. Such devices can be invoked from the circuit netlist, in a similar manner to traditional SPICE-style analog devices. One dimensional and two dimensional devices are supported, with the dimensionality determined by the device model level.

General Form, 1D:	`Z<name> <node> <node> [model name]`
	`[na=<value>] [nd=<value>] [displcur=<value>]`
	`[nx=<value>] [area=<value>] [graded=<value>]`
	`[wj=<value>] [l=<value>] [w=<value>]`
	`[tecplotlevel=<value>] [sgplotlevel=<value>]`
	`[anodebc=<value>] [cathodebc=<value>]`

General Form, 2D:	`Z<name> <node> <node> [node][node] [model name]`
	`[na=<value>] [nd=<value>] [displcur=<value>]`
	`[meshfile=<filename.msh>]`
	`[node1=<electrode name>] [node2=<electrode name>]`
	`[node3=<electrode name>] [node4=<electrode name>]`
	`[node1bc=<neumann, etc.>] [node2bc=<neumann, etc.>]`
	`[node3bc=<neumann, etc.>] [node4bc=<neumann, etc.>]`
	`[x0=<value>] [tecplotlevel=<value>] [sgplotlevel=<value>]`
	`[interpgridsize=<value>]`

Comments:	Most of the PDE parameters are specified on the instance level. At this point the model statement is only used for specifying if the device is 1D or 2D, via the level parameter. The 1D device constructs an evenly spaced mesh, internally, while the 2D device requires a separate mesh file.

Most PDE parameters are specified on the instance level.

Instance parameters	Description	Units	Default	Device Type
	All Levels			
`name`	The instance name must start with a Z.	-	-	1D, 2D
`node`	Minimum of 2 connecting circuit nodes. The 2D device may have as many as 4 nodes, while the 1D device can only have 2.	-	-	1D, 2D
`displcur`	Flag for enabling displacement current	-	0	1D, 2D

Instance parameters	Description	Units	Default	Device Type
area	Cross sectional area of the device.	-	1.0	1D, 2D
tecplotlevel	Quantity of TecPlot output: 0 - no TecPlot files 1 - TecPlot files w/o vector data 2 - TecPlot files with vector data (2D only) 3 - same as 2 but with additional high-resolution interpolated vector output file (2D only) TecPlot files will have the .dat suffix.	-	1	1D, 2D
sgplotlevel	Quantity of sgplot output. 0=no sgplot files. 1=sgplot files. sgplot is a simple plotting program that comes as part of the simgen framework [20]. Mostly, this option is provided in case TecPlot is unavailable. sgplot files will have the *.res suffix.	-	0	1D, 2D
Level 1 (1D) only				
nx	Number of mesh points, x-direction.	-	11	1D
graded	Flag for graded junction vs. abrupt junction. (1=graded, 0=abrupt)	-	0	1D
wj	Junction width.	-	1.0e-4	1D
l, w	Device length and/or width. These parameters mean the same thing for the 1D device.	-	1.0e-3	1D
anodebc	This is a dirichlet boundary condition for the potential at the anode of the device. Specify this boundary condition only if you are running a PDE device with no circuit attached. Under those circumstances, the applied voltage doesn't come from a circuit and you have to provide it. If not provided, it will default to the built in potential.	Volts	Vbi	1D
cathodebc	Same as anodebc, but applied to the cathode instead. The default here is obviously not the built in potential, but rather ground.	Volts	0.0	1D
Level 2 (2D) only				
meshfile	This is a required field for a 2D simulation. Currently, **Xyce** can only read in mesh files created with the simgen framework [20].	-	-	2D

Instance parameters	Description	Units	Default	Device Type
node1, node2, etc.	These are required fields for a 2D simulation. The idea for this parameter is to map a circuit node to a labeled boundary condition from the mesh file. node1 corresponds to the first node of the node list, node2 to the second node, etc. A typical line for this field would be node1=collector.	-	-	2D
node1bc, node2bc, etc.	These are optional fields for a 2D simulation. The idea behind this parameter would be if you wanted to specify a boundary condition at one of the connected nodes that was not dirichlet. The three possibilities are node1bc=dirichlet, node1bc=neumann, and node1bc=mixed. The mixed boundary condition applies a dirichlet condition for electrostatic potential, and a neumann coundition for the two carriers. Generally, it is best to use the default.	-	dirichlet	2D
x0	This is the scaling factor for length. The code will do all of its scaling internally, so it is generally not neccessary to specify it manually. This is provided primarily for testing purposes.	-	max length of device	2D
interpgridsize	This is for interpolated output, and is only relevant if the tecplotlevel is set to 3 or more. In this case, a plot file on an interpolated cartesian mesh is output and this number determines the mesh spacing in the smallest direction. (If the device is wider than it is tall, this is the vertical spacing for the interpolation grid.)	-	20	2D

Table A.36: PDE Device Instance Parameters.

There is only one PDE device model parameter, the level.

Model parameters	Description	Units	Default
LEVEL	The level determines if this is a 1D or a 2D device. 1=1D, 2=2D.	-	1

Table A.37. PDE Device Model Parameters.

B Command Line Arguments

Xyce supports a handful of command line arguments which must be given *before* the netlist filename. While most of these are intended for general use within **Xyce**, others simply give access to new features that, while supported, are not enabled by default but may be in future releases. These options are designated as *trial* options. For example, the `-np` option (not to be confused with the MPI option) enables the new XML-based parser. In a future release, this will be the default parser and this option will no longer be available. The general usage form is as follows:

```
runxyce [arguments] netlist
```

Table B.38 gives a complete lists of supported command line options. In this table, the shaded rows indicate the *trial* options.

Argument	Description	Usage	Default
-h	Help option. Prints usage and exits.	`-h`	-
-np	Use the new version of the netlist parser.	`-np`	-
-delim	Set the output file field.	`-delim <TAB|COMMA|string>`	-
-o	Place the results into specified file.	`-o <file>`	-
-l	Place the log output into specified file.	`-l <file>`	-
-dva	Use faster direct vector access.	`-dva <on|off>`	on
-dma	Use faster direct matrix access.	`-dma <on|off>`	off

Table B.38: List of **Xyce** command line arguments.

The options `-dva` and `-dma` refer to new capabilities in **Xyce** which allow the matrix and vector loads to be performed more quickly. Users may turn off both `-dma` and `-dva`, enable one only, or use them together.

C Runtime Environment

There are two ways to start **Xyce**, using either the `runxyce` (or `runxyce.bat`) for serial **Xyce**, or `xmpirun` for parallel **Xyce**. These scripts set up the run time environment and call the **Xyce** executable. In addition to this, `xmpirun` serves as a wrapper to `mpirun` and `dmpirun`.

Environment variables are used to configure **Xyce**. Users may override the defaults by applying new values. Table C.39 lists the available environment variables and their usage.

Name	Purpose
`COMET_XML_FILE_PATH`	Sets the path to the xml configuration files.
`COMET_DTD_FILE_PATH`	Sets the path to the xml dtd files.
`COMET_UNIT_DICTIONARY_FILE`	Specifies the path and filename of the unit dictionary.

Table C.39. Environment variables used to control **Xyce**.

The version `mpirun` called by `xmpirun` varies with each operating system. Table C.40 shows the actual command executed on the each of the supported platforms. Execution in parallel is highly dependent on how the MPI runtime environment has been set up, and the table only lists the method used when the `mpirun` program has been installed according to the system vendor's defaults.

In general the number of processors is specified by using the `-np` argument to the appropriate `mpirun` command. Some specific considerations are given below.

Running **Xyce** under MPICH

The MPICH implementation of MPI requires that there exist a file of machines on which to run. On RedHat Linux this is installed in /usr/lib/mpich/share. On FreeBSD this is installed in /usr/local/mpich/share. This file must contain one line for each machine on which a process may be started. If you do not have write access to the directory in which the default machines file is stored you may specify an alternate file with the `-machinefile <machinefilename>` option to `mpirun`.

MPICH executes parallel jobs by using the remote shell (rsh) or secure shell (ssh) to the target machine. You may, therefore, be prompted for a password when starting up a multiple processor job.

System	MPI command line
HP/Compaq OSF	`dmpirun -np 2 Xyce_alpha-osf1_MPI anExampleNetlist.cir`
SGI / IRIX 6.5	`mpirun -np 2 Xyce_sgi10k_MPI anExampleNetlist.cir`
Intel X86 / RedHat Linux with MPICH	`mpirun.mpich -np 2 Xyce_linux_MPI anExampleNetlist.cir`
Intel X86 / RedHat Linux with LAM MPI	`mpirun -np 2 Xyce_linux_MPILAM anExampleNetlist.cir`
Intel X86 / FreeBSD with MPICH	`/usr/local/mpich/bin/mpirun -np 2 Xyce_freebsd_MPI anExampleNetlist.cir`
Intel X86 / FreeBSD with LAM MPI	`mpirun -np 2 Xyce_freebsd_MPILAM anExampleNetlist.cir`

Table C.40. Typical command lines for parallel execution.

Running **Xyce** under LAM MPI

Unlike MPICH, LAM MPI requires a daemon process to be running on each machine that will service parallel jobs. This daemon is started by using the `lamboot` program. By default, `lamboot` will run a daemon on the local machine, but it may be given a file name containing a list of machines for multiple-machine jobs. Consult the `bhost` man page for the format of the file.

`lamboot` runs a program called `lamd` which will remain running until it is halted. As long as `lamd` is running you may continue to run parallel jobs. Halt `lamd` using the `lamhalt` command.

Running **Xyce** under HP/Compaq's OSF MPI implementation

On HP/Compaq systems running the OSF operating system the appropriate command to run parallel jobs is `dmpirun`. Its usage is otherwise similar to the MPICH and LAM `mpirun` commands.

D Setting Convergence Parameters for **Xyce**

Because the solution algorithms and methods within **Xyce** are different than those used by other circuit simulation tools (e.g., ChileSPICE), the overall convergence behavior is sometimes different as are the parameters which control this behavior. In general, the current set of defaults in **Xyce** produce solutions which are more accurate than those generated by many similar tools using their defaults. Currently, this may also result in **Xyce** having a longer time-to-solution (transient) since the greater accuracy often requires more time integration steps. However, for a given level of accuracy, **Xyce** will typically be very competitive with existing circuit simulation tools.

One way to reduce the time-to-solution is to reduce the requested accuracy by modifying one or more sets of tolerances from their default values. As this is typically an issue for transient simulations, the following discussion applies to that context only. For a complete list of available solution control options, go to A in Appendix A.

Adjusting Transient Analysis Error Tolerances

By default, **Xyce** uses a BDF-2 time integration method (also known as a 2-step Gear method) for performing transient analysis [21]. This is equivalent to using a Gear method 2 in ChileSPICE (the ChileSPICE default is to use the trapezoid rule). In general, the BDF-2 is a more stable and more accurate method than the trapezoid rule. Also available in **Xyce** is a BDF-1 method (also known as backward Euler or implicit Euler). In **Xyce**, this can be selected via the netlist with an options line:

```
.OPTIONS TIMEINT METHOD=1
```

This will sometimes improve the speed of the code for circuits with highly oscillatory behavior.

Setting RELTOL and ABSTOL

In **Xyce**, there is currently RELTOL and ABSTOL settings for both the time integration package and the nonlinear solver package. Some general guidelines for settings parameters are [21]:

- Use the *same* RELTOL and ABSTOL values for both the TIMEINT and the NONLIN-TRAN .OPTIONS statements.
- For a conservative approach (i.e., safe), set RELTOL=1.0E-(m+1) where m is the desired number of significant digits of accuracy.

- Set ABSTOL to the smallest value at which the solution components (either voltage or current) are essentially insignificant.
- *Note* that the above suggests that ABSTOL < RELTOL.

For a complete list of the time integration parameters, see A.

Adjusting Nonlinear Solver Parameters (in transient mode)

In **Xyce**, the nonlinear solver options for transient analysis are set using the .OPTIONS NONLIN-TRAN line in a netlist. This subsection gives some guidelines for setting this parameters.

- For guidelines on setting RELTOL and ABSTOL, see above.
- RHSTOL – This is the maximum residual error for each nonlinear solution. **Xyce** uses this as a "safety" check on nonlinear convergence. Typically, 1.0E-3 works well.
- DELTAXTOL – This is the weighted update norm tolerance and is the primary check for nonlinear convergence. Since it is weighted (i.e., normalized using RELTOL and ABSTOL), a value of 1.0 would give it the same accuracy as the time integrator. For robustness, the default is 0.33 but sometimes a value of 0.1 may help prevent "time-step too small" errors. A value of 0.01 is considered quite small.
- MAXSTEP – This is the maximum number of Newton (nonlinear) steps for each nonlinear solve. In transient analysis, the default is 40 but can be increased to help prevent "time-step too small" errors. This is roughly equivalent to ITL4 in ChileSPICE.

E Quick Reference for Orcad PSpice Users

This appendix describes the essential differences between **Xyce** and Orcad PSpice with an eye towards providing the ability for those familiar with using PSpice to begin using **Xyce** quickly. Currently, **Xyce** is still in very active state of development and so this section will change as capability is added to **Xyce** in the future.

GUI Support

Probably the most noticeable difference between **Xyce** and PSpice is the lack of a GUI in the current release version of **Xyce**. One, namely ChileCAD, is being developed and is expected to be available for the next release with some basic functionality. Thus, currently the only way to run **Xyce** is from the command line and using standard netlists.

Command Line Options

Command line arguments are supported in **Xyce** but they are different than those of PSPICE. For a complete reference, see Appendix B.

Device Support

In this release version of **Xyce**, not all the devices commonly found in circuit simulation tools are supported. For the complete list, please see the Analog Device Summary in Table 4.2.

Netlist Support

To the extent that specific devices or models are supported in **Xyce**, it supports most of the standard netlist inputs as may be found in standard SPICE. However, the `.OPTIONS` command has several additional features used to expose capabilities specific to **Xyce**. In particular, **Xyce** does not currently support the standard PSpice format `.OPTIONS` line in netlists. Instead, package specific `.OPTIONS` lines are supported according to the following format: `.OPTIONS {PKG} <<TAG=>VALUE> ...` The **Xyce** packages which currently support `.OPTIONS` are:

Global:	`PKG=GLOBAL`
Device Model:	`PKG=DEVICE`
Time Integration:	`PKG=TIMEINT`

Nonlinear Solver:	PKG=NONLIN
Transient Nonlinear Solver:	PKG=NONLIN-TRAN
Linear Solver:	PKG=LINSOL
Parallel Distribution:	PKG=PARALLEL

For a complete description of the supported options, see Appendix A.

Xyce does not support the ".PROBE" statement. Output of Probe format files is done using the ".PRINT" netlist statement. See Appendix A for syntax.

Xyce does not support PSPICE style abbreviations in the ".PRINT" statement. For example, to print out the value of the voltage at node A in a transient simulation you must request .PRINT TRAN V(A), not .PRINT TRAN A.

Converting PSpice ABM Models for Use in **Xyce**

In PSpice, analog behavioral model is achieved by the ABM extensions to its voltage-controlled voltage and current sources, the E and G device types, respectively. The voltage-controlled voltage source provides a voltage output, and the voltage-controlled current source provides a current output. The device's transfer function may contain any mixture of voltages and currents as inputs. Hence, PSpice does not use the F and H current-controlled analogs of the E and G device types in ABM modeling. Rather, the device type is dictated only by the output requirements.

The general forms for the PSpice "E" and "G" ABM extensions are:

```
E<name> <connecting nodes> <ABM keyword> <ABM function>
G<name> <connecting nodes> <ABM keyword> <ABM function>
```

where

<name>	is the device name appended to the E or G device type character
<connecting nodes>	specifies the <(+ node name, - node name)> pair between which the device is connected
<ABM keyword>	specifies the form of the transfer function to be used as one of: **VALUE** arithmetic expression **TABLE** lookup table **LAPLACE** Laplace transform **FREQ** frequency response table **CHEBYSHEV** Chebyshev filter characteristics
<ABM function>	specifies the transfer function as a formula or lookup table as required by the specified <ABM keyword>

Moving a PSpice netlist to **Xyce** requires converting `E` and `G` device types used for ABM modeling to the **Xyce** `B` device type. Note, however, that if an `E` or `G` device is being used within PSpice as a simple linear controlling voltage or current source, that is of the form

```
E<name> <connecting nodes> <controlling nodes> <value>
```

or

```
G<name> <connecting nodes> <controlling nodes> <value>
```

then no conversion is needed since **Xyce** accepts devices of this form.

Xyce `B` device equivalents are only available when the transfer function specified in the PSpice `E` or `G` device is either **VALUE** or **TABLE**. For the **VALUE** function, the **Xyce** `B` device equivalents are

```
B<name> <connecting nodes> V = {<ABM expression>}
```

for the `E` device, and

```
B<name> <connecting nodes> I = {<ABM expression>}
```

for the `G` device. As in PSpice, the expression must be enclosed in braces. Given the following PSpice `G` device

```
G1 2 4 VALUE {V(1)/100}
```

the equivalent **Xyce** `B` device is written

```
B1 2 4 I={V(1)/100}
```

For the **TABLE** transfer function the conversion is similar, the **Xyce** `B` device equivalents are

```
B<name> <connecting nodes> V = {TABLE <ABM expression>}
```

for the `E` device, and

```
B<name> <connecting nodes> I = {TABLE <ABM expression>}
```

for the G device. The form of the ABM expression in this case is:

```
<expression> = <(input value, output value)>*
```

The asterisk indicates there should be one or more (input value, output value) pairs in the table. The table's input values must be in order from lowest to highest. For example, given the following PSpice E device

```
E1 5 0 TABLE V(5) =
+ (0,0)           (.02, 2.69e-3) (.04, 4.10e-3) (.06, 4.62e-3)
+ (.08, 4.46e-3)  (.10, 3.86e-3) (.12, 3.08e-3) (.14, 2.32e-3)
.
.
.
+ (.56, 7.48e-2)  (.58, 1.89e-1) (.60, 4.42e-1)
```

the corresponding **Xyce** B device is written

```
B1 5 0 V={TABLE {V(5)} =
+ (0,0)           (.02, 2.69e-3) (.04, 4.10e-3) (.06, 4.62e-3)
+ (.08, 4.46e-3)  (.10, 3.86e-3) (.12, 3.08e-3) (.14, 2.32e-3)
.
.
.
+ (.56, 7.48e-2)  (.58, 1.89e-1) (.60, 4.42e-1)}
```

F Quick Reference for ChileSPICE Users

A large number of potential **Xyce** users have experience using Sandia's ChileSPICE circuit simulator, which is a shared-memory parallel code based on Berkely's SPICE version 3f5. The following table lists some of the differences between ChileSPICE and **Xyce**.

Issue	Comment
`.SAVE` does not work	**Xyce** does not support this. Use .PRINT instead.
`.probe` does not work	**Xyce** does not support this. Use the FORMAT=PROBE option of .PRINT instead. See Appendix A for syntax.
`.OP` is incomplete	An .OP netlist will run in **Xyce**, but will not produce the extra output normally associated with the .OP statement.
Pulsed source rise time of zero	A requested pulsed source rise/fall time of zero really is zero in **Xyce**. In other simulators, requesting a zero rise/fall time causes them to use the printing interval found on the tran line.
Mutual Inductor Model	Not the same as PSPICE. This is a Sandia developed model.
`.PRINT` line shorthand	Output variables have to be specified as a V(node) or I(source). Listing the node alone will not work.
BSIM3 level	In **Xyce** the BSIM3 level=9. In ChileSPICE the BSIM3 is level=8.
Node names vs. device names	Currently, circuit nodes and devices *MUST* have different names in **Xyce**. Some simulators can handle a device and a node with the same name, but **Xyce** cannot.
Inline comments	**Xyce** does not support semi-colon delimited inline comments.
Interactive mode	**Xyce** does not have an interactive mode.
Time integrator default tolerances	**Xyce** has much tighter default solver tolerances than some other simulators (e.g., ChileSPICE), and thus often takes smaller time steps. As a result, it will often take a greater number of total time steps for a given time interval. To have **Xyce** take time steps comparable to those of ChileSPICE, set the `RELTOL` and `ABSTOL` time integrator options to larger values (e.g., `RELTOL=1.0E-2, ABSTOL=1.0E-6`).
ChileSPICE-specific "operating point voltage sources"	These are not currently supported within **Xyce**.
`.OPTIONS` statements	**Xyce** does not support ChileSPICE style `.OPTION` statements. In **Xyce**, the various packages all (potentially) have their own separate `.OPTIONS` line in the netlist. For a complete description, see Appendix A.

Issue	Comment
DTMAX	**Xyce** does not currently support a maximum time step-size control. The time integration algorithms within **Xyce** use adaptive time-stepping methods that adjust the time-step size according to the activity in the analysis. If the simulator is not providing enough accuracy, the RELTOL and ABSTOL parameters should be decreased for both the time integration package (.OPTIONS TIMEINT) and the transient nonlinear solver package (.OPTIONS NONLIN-TRAN).
Analog Behavioral Models	ChileSPICE supports analog behavioral modeling in the same way as PSpice through the E, F, G and H devices, but **Xyce** supports only the general "B" source. **Xyce**'s E and G sources are purely SPICE 3F5 compatible linear sources. For a complete description, including rules for conversion to the **Xyce** equivalent, see the Appendix E.
Nonlinear Dependent Source (B source) syntax	**Xyce** requires curly braces around all ABM expressions, where ChileSPICE does not. See Appendix A.
.TRAN "UIC" keyword	ChileSPICE requires the use of a keyword UIC on the .TRAN line in order to use initial conditions via IC keywords on instance lines. In **Xyce**, UIC is an unsupported keyword that *should not be specified*. **Xyce** always uses initial conditions specified with IC keywords. Specifying UIC is an error that does not trigger a warning in this version of **Xyce**, but might lead to incorrect behaviors.
Temperature specification	Device temperatures in **Xyce** are specified through the .OPTIONS DEVICE line. ChileSPICE allows a .TEMP line that is not recognized (and is ignored) by **Xyce**.

Table F.41: Incompatibilities with ChileSPICE.

Bibliography

[1] Orcad PSpice User's Guide. Technical report, Orcad, Inc., 1998.

[2] M. S. Eldred, A. A. Giunta, B. G. van Bloemen Waanders, S. F. Wojtkiewicz Jr., W. E. Hart, and M. P. Alleva. DAKOTA, A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. Version 3.0 Reference Manual. Technical Report SAND2001-3796, Sandia National Laboratories, Albuquerque, NM, April 2002.

[3] A. S. Grove. *Physics and Technology of Semiconductor Devices*. John Wiley and Sons, Inc., 1967.

[4] Bruce Hendrickson and Robert Leland. The Chaco User's Guide: Version 2.0. Technical Report SAND94–2692, Sandia National Laboratories, Albuquerque, NM, December 1994.

[5] H. A. Watts, E. R. Keiter, S. A. Hutchinson, and R. J. Hoekstra. Time integration for the Xyce parallel electronic simulator. In *ISCAS 01*, October 2000.

[6] T. Banwell. Bipolar transistor circuit analysis using the lamber w-function. *IEEE Transactions on Circuits and Systems*, 47(11):1621–1633, November 2000.

[7] R. S. Tuminaro, M. A. Heroux, S. A. Hutchinson, and J. N. Shadid. Official Aztec User's Guide: Version 2.1. Technical report, Sandia National Laboratories, Albuquerque, NM, December 1999.

[8] T. A. Fjeldly, T. Ytterdal, and M. Shur. *Introduction to Device Modeling and Circuit Simulation*. Wiley InterScience, 1998.

[9] T. A. Fjeldly, Yanqing Deng, and M. Shur. Physics-based modeling of high intensity ionizing radiation effects in bipolar junction transistors. Technical report, 2000.

[10] P. Antognetti and G. Massobrio. *Semiconductor Device Modeling with SPICE*. McGraw-Hill, 1988.

[11] J.H. Huang, Z.H. Liu, M.C. Jeng, K. Hui, M. Chan, P.K. KO, and C. Hu. BSIM3 Manual. Technical report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.

[12] A. Vladimirescu and S. Lui. The Simulation of MOS Integrated Circuits using SPICE2. Technical Report Memorandum No. M80/7, February 1980.

[13] H. Shichman and D. A. Hodges. Modeling and simulation of insulated-gate field-effect transistor switching circuits. *IEEE Journal of Solid-State Circuits*, SC-3:285, September 1968.

[14] B. J. Sheu, D. L. Scharfetter, P.-K. Ko, and M.-C. Jeng. Bsim: Berkeley short-channel igfet model for mos transistors. *IEEE Journal of Solid-State Circuits*, SC-22:558–566, August 1987.

[15] J. R. Pierret. A MOS Parameter Extraction Program for the BSIM Model. Technical Report Memorandum No. M84/99 and M84/100, November 1984.

[16] Ping Yang, Berton Epler, and Pallab K. Chatterjee. An investigation of the charge conservation problem for mosfet circuit simulation. *IEEE Journal of Solid-State Circuits*, SC-18(1), February 1983.

[17] BSIM3v3.1 Manual. Technical report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.

[18] J. C. Bowers and H. A. Neinhaus. SPICE2 Computer Models for HEXFETs. Technical Report Application Note 954A, International Rectifier Corporation, Berkeley, CA 94720.

[19] M. Bucher, C. Lallement, C. Enz, F. Theodoloz, and F. Krummenacher. The EPFL/EKV MOSFET Model Equations for Simulation Technical Report: Model Version 2.6. Technical report, Electonics Laboratories, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, September 1997.

[20] K. M. Kramer and W. N. G. Hitchon. *Semiconductor Devices: A Simulation Approach*. Prentice-Hall, 1997.

[21] K.E. Brenan, S.L Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, 1996.

Index

Xyce
- ABSTOL, 153
- RELTOL, 153
- convergence, 153
- running, 53
- time-to-solution, 153

.FUNC, 79
.INC, 79
.OPTIONS, 79
.PARAM, 79
.PRINT, 73, 77
.SUBCKT, 74
ABSTOL, 82, 85, 87
BPENABLE, 84
CONSTSTEP, 82
DIRECTLINSOLV, 86, 87
EXITSTEP, 84
EXITTIME, 84
MAXSTEP, 85, 87
OUTPUT, 64
RELTOL, 82, 85, 87
.PRINT, 68, 69
.TRAN, 89
OUTPUT, 55
RESTART, 55, 64, 89

abrupt junction, 109
AC analysis, 52
accurate, 153
algorithm
- time integration, 52, 160

analysis
- AC, 52
- control parameters, 79
- creating and running, 51
- creation, 52
- DC, 23, 52, 57, 85
- DC sweep, 29
- DC sweep, 52
- DC sweep, 58
- options, 79
- transient, 23, 24, 30, 52, 61, 62, 73, 85

Aztec, 88

behavioral model, 4, 39
- analog behavioral modeling (ABM), 49
- analog behavioral modeling (ABM), 50
- analog behavioral modeling (ABM), 49

bias point, 58, 62, 73
BJT, 121
- model parameters, 123
- operating temperature, 121

bsource, 120

capacitor, 93
- age-aware, 93, 94
- model parameters, 94

Chaco, 54, 55
checkpoint, 64, 89
- format, 64, 89

ChileCAD, 155
ChileSPICE, 18, 160
circuit
- elements, 36
- simulation, 36
- topology, 36, 37

command line, 29, 53, 149
- arguments, 149
- output, 54

convergence, 153

DAKOTA, 23, 24

DC analysis, 52, 57, 85
DC Operating Point, 24
DC Operating Point, 23
DC Sweep, 58
DC sweep, 29, 52
  OP Analysis, 58
  running, 58
device
  B (nonlinear dependent) source, 50, 157
  ABM device
    PSpice equivalent, 156
  analog, 38, 39, 91
  analog device summary, 40
  B source, 40
  behavioral, 50
  behavioral model, 16, 22, 39
  bipolar junction transistor (BJT, 40
  BJT, 121
  bsource, 120
  capacitor, 40, 93
  device types, 39
  diode, 40, 102
  equations, 91
  independent current source, 40, 114
  independent voltage source, 40, 111
  inductor, 40, 96
  instance, 39
  lossless transmission line, 144
  model selections, 80
  MOSFET, 40, 126
  mutual inductor, 40
  mutualinductor, 98
  nonlinear dependent source, 40
  package, 22
  PDE, 22
  PDE Devices, 40, 146
  resistor, 40, 100
  specifying ABM devices, 50
  subcircuit, 40
  table, 157
  transmission line, 40
  voltage controlled current source, 40, 119
  voltage controlled switch, 40, 142
  voltage controlled voltage source, 40, 118
diode, 102
  level=3 model parameters, 107
  model parameters, 103
  operating temperature, 102

Example
  checkpointing, 64
  circuit construction, 26
  DC sweep, 29
  declaring parameters, 41
  restarting, 65
  subcircuit model definition, 47, 48
  transient analysis, 32
  using expressions, 42
  using parameters, 41

graph partitioning, 54
ground nodes, 37
GUI, 29, 53, 155

independent current source, 114
independent voltage source, 111
inductor, 96
  model parameters, 97
initial conditions, 52

Lambert-W Function, 81
lossless transmission line, 144
  model parameters, 144

Microsoft Windows, 53
model, 91
  definition, 46, 74
  model organization, 48
MOSFET, 126
  model parameters, 136
MPI, 29, 53
mutualinductor, 98
  model parameters, 99

netlist, 26, 36
  .END, 36
  .END statement, 27
  analog devices, 38
  arithmetic expressions, 44

command elements, 38
commands, 72
comment, 91
comments, 27, 37
converting from PSpice, 157
device description, 38
element, 36
end line, 37
expression operators, 43
expressions, 41
in-line comment, 91
line continuation, 91
model definition, 38
node names, 37
parameters, 39
reference, 71
restart, 64
scaling factors, 36
sources, 62
subcircuit, 38
title, 27
title line, 36, 37
using expressions, 41
node names, 37

Object-oriented, 16
OP analysis, 58
Operating Point, 23, 24, 72
output
control, 77
distributed voltage sources, 55
distributed voltage nodes, 55
time values, 63

parallel
communication, 54
computing, 15, 16, 21, 22
distributed-memory, 16, 22
efficiency, 16, 22, 54
graph partitioning, 54
graph partitioning, 54
large scale, 22
load balance, 54
message passing, 16, 22
MPI, 29, 53
number of processors, 53
options, 89
shared-memory, 16, 22
parameters
convergence, 153
PDE Devices, 40, 146
instance parameters, 148
model parameters, 148
time integration parameters, 82
platforms
Compaq/OSF, 54, 152
Intel X86/FreeBSD, 54, 152
Intel X86/Linux, 54, 152
Intel X86/Microsoft Windows 2000, 54
SGI/IRIX, 54, 152
Sun/Solaris, 54
PSpice, 18, 26, 50, 155–157
E device, 156, 157
G device, 156
G device, 157
Probe, 70, 78

resistor, 100
model parameters, 100
restart, 55, 64, 89
file, 90
format, 64, 90
results
evaluating, 70
output control, 77
output control, 68
output frequency, 68
output options, 67, 89
print, 73
print commands, 69
running **Xyce**, 53
runxyce, 29, 53

Sandia National Laboratories, 15
schematic capture, 26
simulation
analog, 22, 23
device, 23
digital, 22, 23
mixed signal, 22, 23
solution time, 153
solvers

control parameters, 79
iterative linear, 55
linear
Aztec, 88
iterative (preconditioned Krylov methods), 88
options, 88, 89
sparse-direct, 88
Trilinos, 88
nonlinear
options, 85, 86, 154
nonlinear-transient
options, 86, 88
time integration
options, 153
time integration, 73, 160
options, 82, 84
transient, 63
sources, 62
defining time-dependent, 62
time-dependent, 63
waveforms, 62
SPICE, 26, 36
subcircuit, 74, 91, 145
designation, 76
hierarchy, 47
name, 75
nesting, 76
node zero, 76
scope, 47
scoping, 76

table, 157
time step, 52
maximum size, 63
size, 63, 73, 160
topology, 37
transient analysis, 30, 52, 61, 62, 85
error tolerances, 153
Trilinos, 88

Unix, 18

voltage controlled current source, 119
voltage controlled voltage source, 118
voltage controlled switch, 142
model parameters, 142

Windows, 53

xmpirun, 29, 53

ZOLTAN, 54, 55

DISTRIBUTION:

1 Steven P. Castillo
Klipsch School of Electrical and Computer Engineering
New Mexico State University
Box 3-o
Las Cruces, NM 88003

1 Kwong T. Ng
Klipsch School of Electrical and Computer Engineering
New Mexico State University
Box 3-o
Las Cruces, NM 88003

1 Nick Hitchon
Electrical and Computer Engineering
University of Wisconsin
1415 Engineering Drive
Madison, WI 53706

1 Mark Kushner
Department of Electrical and Computer Engineering
University of Illinois
1406 W. Green Street
Urbana, IL 61801

1 Andrew J. Christlieb
Department of Mathematics
University of Michigan
2470 East Hall
Ann Arbor, MI 48109

1 Ron Kielkowski
RCG Research, Inc
8605 Allisonville Rd, Suite 370
Indianapolis, In 46250

1 Mike Davis
Software Federation, Inc.
211 Highview Drive
Boulder, Co 80304

1 Wendland Beezhold
Idaho Accelerator Center
1500 Alvin Ricken Drive
Pocatello, Idaho 83201

1 Kartikeya Mayaram
Department of Electrical and Computer Engineering
Oregon State University
Corvallis, OR 97331-3211

1 Linda Petzold
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106-5070

1 Jaijeet Roychowdhury
4-174 EE/CSci Building
200 Union Street S.E.
University of Minnesota
Minneapolis, MN 55455

1 C.-J. Richard Shi
VLSI and Electronic Design Automation
210 EE/CSE Bldg.
Box 352500
University of Washington
Seattle, WA 98195

1 Homer F. Walker
WPI Mathematical Sciences
100 Institute Road
Worcester, MA 01609

1 Dan Yergeau
CISX 334
Via Ortega
Stanford, CA 94305-4075

1 Masha Sosonkina
319 Heller Hall
10 University Dr.
Duluth, MN 55812

1 Misha Elena Kilmer
113 Bromfield-Pearson Bldg.
Tufts University
Medford, MA 02155

1 Tim Davis
P.O. Box 116120
University of Florida
Gainesville, FL 32611-6120

1 Achim Basermann
C&C Research Laboratories, NEC Europe Ltd.
Rathausallee 10
D-53757 Sankt Augustin
Germany

1 Philip A. Wilsey
Experimental Computing Laboratory
Department of Electrical & Computer Engineering and Computer Science
College of Engineering
P.O. Box 210030
University of Cincinnati
Cincinnati, Ohio 45221-0030

1 Dale E. Martin
Clifton Labs
3678 Fawnrun Dr.
Cincinnati, OH 45241

1 MS 0151
Tom Hunter, 09000

1 MS 0513
Al Romig, 01000

1 MS 0457
John Stichman, 02000

1 MS 0321
Bill Camp, 09200

1 MS 0841
Thomas C. Bickel, 09100

1 MS 1079
Marion Scott, 01700

1 MS 9003
Kenneth E. Washington, 08900

1 MS 0318
Paul Yarrington, 09230

1 MS 1071
Mike Knoll, 01730

1 MS 0310
Robert Leland, 09220

1 MS 0316
Sudip Dosanjh, 09233

1 MS 0525
Paul V. Plunkett, 01734

1 MS 0835
J. Michael McGlaun, 09140

1 MS 0835
Steven N. Kempka, 09141

1 MS 0826
John D. Zepper, 09143

1 MS 0824
Jaime L. Moya, 09130

1 MS 0828
Martin Pilch, 09133

1 MS 0139
Stephen E. Lott, 09905

1 MS 0310
Mark D. Rintoul, 09212

1 MS 1110
David Womble, 09214

1 MS 1111
Bruce Hendrickson, 09215

1 MS 1110
Neil Pundit, 09223

1 MS 1110
Doug Doerfler, 09224

1 MS 0822
Philip Heermann, 09227

1 MS 0819
Edward Boucheron, 09231

1 MS 0820
Patrick Chavez, 09232

1 MS 0316
John Aidun, 09235

10 MS 0316
Scott A. Hutchinson, 09233

1 MS 0316
Eric R. Keiter, 09233

1 MS 0316
Robert J. Hoekstra, 09233

1 MS 0316
Joseph P. Castro, 09233

1 MS 0316
David R. Gardner, 09233

1 MS 0316
Gary Hennigan, 09233

1 MS 0316
Roger Pawlowski, 09233

1 MS 0316
Richard Schiek, 09233

1 MS 1111
John N. Shadid, 09233

1 MS 1111
Andrew Salinger, 09233

1 MS 0847
Scott Mitchell, 09211

1 MS 0847
Mike Eldred, 09211

1 MS 0847
Tim Trucano, 09211

1 MS 0847
Bart van Bloemen Waanders, 09211

1 MS 0196
Elebeoba May, 09212

1 MS 1110
Todd Coffey, 09214

1 MS 1110
David Day, 09214

1 MS 1110
Mike Heroux, 09214

1 MS 1110
James Willenbring, 09214

1 MS 1111
Karen Devine, 09215

1 MS 0310
Jim Ang, 09220

1 MS 1109
Robert Benner, 09224

1 MS 0822
Pat Crossno, 09227

1 MS 0822
David Rogers, 09227

1 MS 0316
Harry Hjalmarson, 09235

1 MS 0525
Steven D. Wix, 01734

1 MS 0525
Thomas V. Russo, 01734

1 MS 0525
Lon Waters, 01734

1 MS 0525
Regina Schells, 01734

1 MS 0525
Carolyn Bogdan, 01734

1 MS 0525
Mike Deveney, 01734

1 MS 0525
Raymond B. Heath, 01734

1 MS 0525
Ronald Sikorksi, 01734

1 MS 0525
Albert Nunez, 01734

1 MS 1081
Paul E. Dodd, 01762

1 MS 0660
Roger F. Billau, 09519

1 MS 0874
Robert Brocato, 01751

1 MS 1081
Charles E. Hembree, 01739

1 MS 0311
Greg Lyons, 02616

1 MS 0311
Martin Stevenson, 02616

1 MS 0328
Fred Anderson, 02612

1 MS 0537
Perry Molley, 02331

1 MS 0537
Siviengxay Limary, 02331

1 MS 0537
John Dye, 02331

1 MS 0537
Barbara Wampler, 02331

1 MS 0537
Doug Weiss, 02333

1 MS 0537
Scott Holswade, 02333

1 MS 0481
Joel Brown, 02132

1 MS 0405
Todd R. Jones, 12333

1 MS 0405
Thomas D. Brown, 12333

1 MS 0405
Donald C. Evans, 12333

1 MS 9101
Rex Eastin, 08232

1 MS 9101
Seung Choi, 08235

1 MS 9409
William P. Ballard, 08730

1 MS 9202
Kathryn R. Hughes, 08205

1 MS 9202
Rene L. Bierbaum, 08205

1 MS 9202
Kenneth D. Marx, 08205

1 MS 9202
Stephen L. Brandon, 08205

1 MS 9217
Stephen W. Thomas, 08950

1 MS 9217
Tamara G. Kolda, 08950

1 MS 9217
Kevin R. Long, 08950

1 MS 1179
Leonard Lorence, 15341

1 MS 1179
David E. Beutler, 15341

1 MS 1179
Brian Franke, 15341

1 MS 0835
Randy Lorber, 09141

1 MS 1152
Mark L. Kiefer, 01642

1 MS 9018
Central Technical Files, 8945-1

2 MS 0899
Technical Library, 9616

1 MS 0612
Review & Approval Desk, for DOE/OSTI, 9612